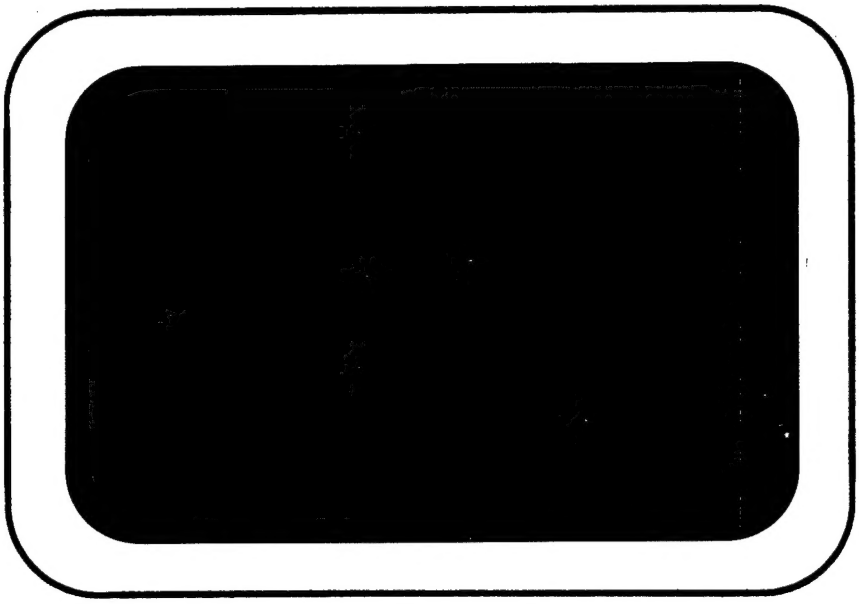
University of Washington



Aerospace and Energetics Research Program

REPORT DOCUMENTATION PAGE

AFRL-SR-BL-TR-00-

0654

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AN
	99/2/23	Final Progress Report 98/5/1 - 98/11/30

4. TITLE AND SUBTITLE
"DEVELOPMENT OF AN ADVANCED IMPLICIT ALGORITHM FOR MHD COMPUTATIONS ON PARALLEL SUPERCOMPUTERS"

5. FUNDING NUMBERS
F49620-98-1-0395

6. AUTHOR(S)
SHUMLAK, Uri

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Washington
Grant & Contract Services
3935 University Way NE
Seattle, WA 98105-6613

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Naval Research (ONR)
1107 NE 45th Street, Suite 350
Seattle, WA 98105-4631

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

20001205 079

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
unclassified/unlimited

DISTRIBUTION STATEMENT
Approved for Public Release
Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***

The primary objective of this project is to develop an advanced algorithm for parallel supercomputers to model time-dependent magnetohydrodynamics (MHD) in all three dimensions. This will provide a valuable tool for the design and testing of plasma related technologies that are important to the Air Force and industry. Implementing the algorithm on parallel supercomputers will allow the detailed modeling of realistic plasmas in complex three-dimensional geometries.

We have developed a time-dependent, two-dimensional, arbitrary-geometry version of the algorithm, placed it into a testbed code, added the modifications necessary for viscous and resistive effects, and tested the code against known analytical problems. We have implemented the algorithm on a parallel architecture and investigated parallelization strategies. Future plans include installing the algorithm into MACH2, optimizing the parallelization, extending the code to three dimensions, installing the three-dimensional algorithm into MACH3, and calibrating the code with experimental data.

14. SUBJECT TERMS	15. NUMBER OF PAGES
implicit algorithm, parallel computer, lower-upper symmetric-Gauss-Siedel, LUSGS, approximate Riemann solver, magnetohydrodynamic, MHD, Hartmann flow	81
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
unclassified	unclassified	unclassified	unclassified/unlimited

Contents

1 Executive Summary 1

2 Project Description 1
- 2.1 Research Objectives . . . 2
- 2.2 Technical Description . . . 2
  - 2.2.1 MHD Plasma Model . . . 2
  - 2.2.2 Algorithm Overview . . . 4
  - 2.2.3 LU-SGS Relaxation Scheme . . . 7
  - 2.2.4 More Powerful Relaxation Schemes . . . 8
  - 2.2.5 Approximate Riemann Solver . . . 9
  - 2.2.6 General Multiblock Implementation . . . 10
  - 2.2.7 Second-Order Accurate Boundary Conditions . . . 12
  - 2.2.8 Unaligned Finite Volumes . . . 14
  - 2.2.9 Parabolic MHD Terms . . . 16
  - 2.2.10 Hall Effect Physics . . . 20

3 Benchmarks and Applications 22
- 3.1 Ideal MHD Test Problems . . . 22
  - 3.1.1 One-Dimensional Coplanar MHD Riemann Problem . 22
  - 3.1.2 Oblique Shock . . . 26
- 3.2 Viscous and Resistive MHD Test Problems . . . 28
  - 3.2.1 Laminar Flow . . . 29
  - 3.2.2 Resistive Diffusion . . . 30
  - 3.2.3 Hartmann Flow . . . 32
- 3.3 MPD Plasma Thruster . . . 35
- 3.4 Magnetic Reconnection . . . 41
  - 3.4.1 Problem Description . . . 44
  - 3.4.2 Linear Analysis . . . 45
  - 3.4.3 Non-Linear Analysis . . . 47
- 3.5 HIT Injector . . . 47
- 3.6 Three-Dimensional Magnetic Relaxation . . . 50
- 3.7 Nonlinear Tilt Instability in the Spheromak . . . 52

4 Parallel Computer Implementation 58
- 4.1 Fine-Grain Parallelization . . . 58
- 4.2 Coarse-Grain Parallelization . . . 60
  - 4.2.1 Domain Decomposition . . . 60
  - 4.2.2 Implementation of the Patch Decomposition . . . 62

4.2.3 Message Passing 63
4.2.4 Load Balancing 63
4.2.5 Results 63
4.3 Parallel Performance on Windows NT 66

5 Professional Interactions **67**
5.1 Project Personnel 67
5.2 Collaborations 69
5.2.1 Air Force Research Laboratory 69
5.2.2 Sandia National Laboratories 69
5.2.3 University of Washington 69
5.3 Publications 69
5.4 Presentations 70

6 Conclusions **70**

List of Figures

1 Top view of a non-simply connected grid. (Only the block boundaries are shown for clarity.) 11
2 Hierarchy of the derived data types. The most basic derived data structure is the cell. 13
3 Schematic drawing of a finite volume cell with local coordinate systems at its interfaces. The fluxes are only needed along the $\mathbf{x_i}$ directions. 15
4 Rotation of a cell coordinate system that aligns x^o with the normal to the cell face $\hat{n}$. 16
5 Plot of a one dimensional shock simulation (pressure) using the new solver and the old solver. Notice the remnant discontinuity shock is reduced by the new solver. 17
6 Plot of a one dimensional simulation of double rarefaction waves using the new solver. The pressure in the wake of the rarefaction waves continues to decrease towards zero without ever becoming negative. 18
7 Positioning for (a) vertex-centered and (b) face-centered parabolic fluxes. The face-centered fluxes produced more accurate and stable solutions. 19
8 Dispersion relation for three values of the Hall parameter, $\omega_{ce}\tau_e$. A Hall parameter of zero corresponds to ideal MHD. . 21
9 Hall effect induced motion of the magnetic field, $k = 5.0$ $\omega_{ci}\tau_A = 3.5$. The lower series of plots is a simulation without the Hall effect. Shown are contours of out of plane magnetic field intensity. 23
10 Numerical solution of coplanar Riemann problem. Density profile is shown initially and after solution has evolved for 400 time steps. 24
11 Numerical solution of coplanar Riemann problem. Transverse magnetic field profile is shown initially and after solution has evolved for 400 time steps. 25
12 Comparison of numerical and exact solution of coplanar Riemann problem for $B_x = 0$ case. 26
13 Geometry of oblique shock test problem. 27
14 Density contours and field lines for an $M = 3$ flow impinging on a perfectly conducting plate at an angle of 25 degrees. . . 27

15 Logarithm of the two-norm of the energy equation residual plotted as a function of iteration number for explicit and implicit solutions of channel flow with horizontal velocity and vertical magnetic field imposed at the left boundary. 28
16 Simulation of laminar flow between parallel plates in the presence of a constant pressure gradient. The velocity profile is parabolic as expected from the analytical solution. 30
17 The Hartmann flow geometry showing the moving parallel plates and the cross magnetic field. 31
18 Hartmann flow simulation with $H = 10^4$. Flow velocity vectors and magnetic field lines are shown. The velocity of the flow is zero everywhere except at the plates. The magnetic field lines have a constant slope through the domain. 33
19 Hartmann flow simulation with $H = 0.1$. Flow velocity vectors and magnetic field lines are shown. The velocity profile is linear and the magnetic field lines have an "S" shape caused by the bulk fluid flow. 34
20 Hartmann flow simulation with $H = 10$. Flow velocity vectors and magnetic field lines are shown. The flow velocity only exists close to the plates. The magnetic field lines are linear around the midplane. 35
21 Geometry of the two-dimensional MPD thruster. 36
22 Plasma velocity as a function of x and time for explicit time differencing simulation and dual time-stepping (implicit) simulation. 37
23 Magnetic field as a function of x and time for explicit time differencing simulation and dual time-stepping (implicit) simulation. 38
24 Velocity vectors for the MPD thruster. The plasma is accelerated down the gun by the $\mathbf{I} \times \mathbf{B}$ force and a boundary layer develops. The internal blocks illustrate the decomposition of the domain used for the validation of the parallel version of the code. 39
25 Magnetic field (B_z) contours for the MPD thruster. The gradient in the magnetic field produces the force applied to the plasma. 40
26 (a) Four block grid for the axisymmetric MPD simulation and (b) the initial contours of the magnetic field. (Black contours represent higher values.) 42

27 (a) Flow field and (b) contours of azimuthal magnetic field. (Black contours represent higher values.) Note the vortex ring shedding off the cathode. 43
28 Schematic of planar sheet pinch problem [from H. P. Furth, Phys. Fluids **28**(6), 1595 (1985)]. 44
29 Equilibrium profiles of normalized magnetic field and resistivity. 45
30 The eigenfunctions for $Lu = 10^3$ and $\alpha = 0.5$. 46
31 The linear and non-linear evolution of the reconnected flux. . 47
32 Flux contours of the developed non-linear instability. 48
33 Schematic of the HIT plasma experiment. 48
34 Results of two-dimensional simulation of HIT injector. Plot shows density contours and poloidal magnetic field lines. . . . 49
35 Contours of poloidal flux showing the toroidal mode structure of a relaxing compact toroid with a magnetic Reynolds number of 10^4 after 10 Alfvén transit times. 51
36 Contours of poloidal flux showing only a weak toroidal mode structure for a relaxing compact toroid with a magnetic Reynolds number of 10^3 after 10 Alfvén transit times. 52
37 The ten block grid used for the spheromak simulation. Some of the blocks has been removed for illustration. 54
38 Two alternate grids that can be used for simulations of cylindrical configurations. **(a)** The pie slice grid has large aspect ratio cells along circumference and a singularity at the axis. **(b)** The distorted square grid has high aspect ratio cells along the circumference. 54
39 Initial velocity field. Contours represent the magnitude of the toroidal velocity component and the vectors represent the poloidal velocity. Velocity field is normalized with respect to the Alfvén speed. 55
40 Contours of toroidal magnetic field B_θ through a cross-section of the spheromak at $\theta = 90^o$. White contours represents positive values and black negative values. 56
41 Evolution of the kinetic energy of the spheromak as a function of normalized time. The solid line is the linear growth rate. . 57
42 The 20×20 lower (a) and upper (b) tridiagonal block matrices for the LU-SGS algorithm with a grid of 4×5 cells. 58
43 The parallel speedup for a problem with constant size grid using a fine-grain parallelization approach. 59
44 (a) Strip decomposition and (b) patch decomposition of a 2-D domain. 61

45 Column decomposition of a 3-D domain is an immediate extension of the patch decomposition of 2-D domains. 62
46 Fixed grid (400×80) speedup results. Note the superlinear speedup of the explicit mode. 64
47 Scaled grid (50×10 per processor) speedup results. 66
48 Screen shot of the GUI which controls our code on Windows NT platforms. 68

1 Executive Summary

The primary objective of this project is to develop an advanced algorithm for parallel supercomputers to model time-dependent and steady state magnetohydrodynamics (MHD) in all three dimensions. A viable time-dependent, three-dimensional MHD code will provide a valuable tool for the design and testing of plasma related technologies that are important to the Air Force and industry. These applications include portable pulsed power, high power microwave devices, advanced plasma thrusters for space propulsion, hypersonic drag reduction, nuclear weapons effects simulations, radiation production for counter proliferation, and fusion for power generation. Several of these technologies are specifically mentioned in the *New World Vistas* Report from the USAF Scientific Advisory Board.[1] Implementing the algorithm on parallel supercomputers will allow the detailed modeling of realistic plasmas in complex three-dimensional geometries.

We have developed a time-dependent, three-dimensional, arbitrary-geometry MHD algorithm with viscous and resistive effects and tested the code against known analytical problems. We have implemented the algorithm on a parallel architecture and optimized the parallelization strategy. The algorithm has been cast using unaligned finite volumes, instead of generalized coordinates, which has greatly improved the accuracy of the code. Global second-order accuracy was achieved by using second-order boundary conditions for both internal (interblock) and external (physical) boundaries. Future plans include investigating more powerful implicit solvers, extending the code to model multi-temperature effects including the presence of neutral gas molecules.

As a result of this project several professional collaborations now exist between the Department of Aeronautics and Astronautics at the University of Washington and the Air Force Research Laboratory, Lawrence Livermore National Laboratory, the University of Michigan, the University of Colorado, and other departments at the University of Washington. The work from this project has been presented at international conferences and published in a refereed journal.

2 Project Description

Plasmas are essential to many technologies that are important to the Air Force, many of which have dual-use potential. These applications include portable pulsed power, high power microwave devices, hypersonic drag re-

duction, advanced plasma thrusters for space propulsion, nuclear weapons effects simulations, radiation production for counter proliferation, and fusion for power generation. In general, plasmas fall into a density regime where they exhibit both collective (fluid) behavior and individual (particle) behavior. Many plasmas of interest can be modeled by treating the plasma like a conducting fluid and assigning macroscopic parameters that accurately describe its particle-like interactions. The magnetohydrodynamic (MHD) model is a plasma model of this type.

2.1 Research Objectives

The objectives of the project are to:

- Develop a coupled, implicit, time-accurate algorithm for three-dimensional, viscous, resistive MHD simulations for time-dependent and steady state solutions;
- Integrate the algorithm into the MACH3 code,[2] which was developed at the Air Force Research Laboratory;
- Validate the code with analytical and experimental data; and
- Apply the code to analyze plasma experiments at the Air Force Research Laboratory [the magnetic flux compression generator (MCG) experiments and the liner implosion system (WFX)[3]] and at the University of Washington [Helicity Injected Tokamak (HIT)[4]].

2.2 Technical Description

2.2.1 MHD Plasma Model

The three-dimensional, viscous, resistive MHD plasma model is a set of mixed hyperbolic and parabolic equations. The Navier-Stokes equations are also of this type. This project applies some advances that have been made in implicit algorithms for the Navier-Stokes equations to the MHD equations. These implicit algorithms solve the equation set in a fully coupled manner, which generates better accuracy than the current methods used for MHD simulations.

When expressed in conservative, non-dimensional form, the MHD model

is described by the following equation set.

$$\frac{\partial}{\partial t}\begin{bmatrix}\rho\\ \rho\mathbf{v}\\ \mathbf{B}\\ e\end{bmatrix}+\nabla\cdot\begin{bmatrix}\rho\mathbf{v}\\ \rho\mathbf{v}\mathbf{v}-\mathbf{B}\mathbf{B}+(p+\mathbf{B}\cdot\mathbf{B}/2)\,\bar{\bar{\mathrm{I}}}\\ \mathbf{v}\mathbf{B}-\mathbf{B}\mathbf{v}\\ (e+p+\mathbf{B}\cdot\mathbf{B}/2)\,\mathbf{v}-(\mathbf{B}\cdot\mathbf{v})\,\mathbf{B}\end{bmatrix}=$$

$$\nabla\cdot\begin{bmatrix}0\\ (ReAl)^{-1}\,\bar{\bar{\tau}}\\ (RmAl)^{-1}\,\bar{\bar{\Xi}}(\bar{\bar{\eta}},\mathbf{B})\\ (ReAl)^{-1}\,\mathbf{v}\cdot\bar{\bar{\tau}}-(RmAl)^{-1}\,\bar{\bar{\eta}}\cdot(\nabla\times\mathbf{B})\times\mathbf{B}+\frac{M_i}{2}\,(PeAl)^{-1}\,\bar{\bar{k}}\cdot\nabla T\end{bmatrix} \quad (1)$$

The variables are density (ρ), velocity ($\mathbf{v}$), magnetic induction ($\mathbf{B}$), pressure (p), energy density (e), and temperature (T). $\bar{\bar{\Xi}}(\bar{\bar{\eta}},\mathbf{B})$ is the transverse resistive electric field tensor which is described in Section 2.2.9. M_i is the ion mass. The energy density is

$$e=\frac{p}{\gamma-1}+\rho\frac{\mathbf{v}\cdot\mathbf{v}}{2}+\frac{\mathbf{B}\cdot\mathbf{B}}{2} \quad (2)$$

where $\gamma = c_p/c_v$ is the ratio of the specific heats. The non-dimensional tensors are the stress tensor ($\bar{\bar{\tau}}$), the electrical resistivity ($\bar{\bar{\eta}}$), and the thermal conductivity ($\bar{\bar{k}}$), and $\bar{\bar{\mathrm{I}}}$ is the identity matrix. The non-dimensional numbers are defined as follows:

$$\begin{array}{lll}\text{Alfvén Number :} & Al & \equiv V_A/V\\ \text{Reynolds Number :} & Re & \equiv LV/\nu\\ \text{Magnetic Reynolds Number :} & Rm & \equiv \mu_o LV/\eta\\ \text{Péclet Number :} & Pe & \equiv LV/\kappa\end{array} \quad (3)$$

The characteristic variables are length (L), velocity (V), Alfvén speed ($V_A = B/\sqrt{\mu_o\rho}$), kinematic viscosity (ν), electrical resistivity (η), and thermal diffusivity ($\kappa = k/\rho c_p$). μ_o is the permeability of free space ($4\pi\times10^{-7}$).

For convenience, the MHD equation set [eqn(1)] is rewritten in the following compact form

$$\frac{\partial Q}{\partial t}+\nabla\cdot\bar{\bar{T}}_h=\nabla\cdot\bar{\bar{T}}_p, \quad (4)$$

where Q is the vector of conservative variables, $\bar{\bar{T}}_h$ is the tensor of hyperbolic fluxes, and $\bar{\bar{T}}_p$ is the tensor of parabolic fluxes. The forms of these vectors and tensors can be seen from eqn(1). The hyperbolic fluxes are associated with wave-like motion, and the parabolic fluxes are associated with diffusion-like motion.

2.2.2 Algorithm Overview

Because of the natural differences between hyperbolic and parabolic equations, the methods for solving them are very different. Since the MHD equations are of mixed type the hyperbolic and parabolic terms must be handled differently. The hyperbolic fluxes are differenced by applying an implicit, approximate Riemann algorithm that properly accounts for their wave-like behavior. The parabolic terms are discretized by applying explicit central differencing.

The design of the overall algorithm is primarily driven by the numerical techniques that must be used to discretize the hyperbolic terms. Therefore, we begin by considering the ideal MHD equations, which are obtained from eqn(4) by setting all the parabolic terms ($\bar{\bar{T}}_p$) to zero.

In one dimension they are

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} = \frac{\partial Q}{\partial t} + A\frac{\partial Q}{\partial x} = 0, \tag{5}$$

where F is the flux vector in the x direction (i.e. $\bar{\bar{T}}_h = (F, G, H)$) and A is its Jacobian.

$$A = \frac{\partial F}{\partial Q} \tag{6}$$

Here, Q is the vector of conserved variables:

$$Q = (\rho, \rho v_x, \rho v_y, \rho v_z, B_y, B_z, e)^T . \tag{7}$$

This is a set of hyperbolic equations and thus A has a complete set of real eigenvalues given by

$$\lambda = (v_x, v_x \pm V_{fast}, v_x \pm V_{slow}, v_x \pm V_{Ax})^T , \tag{8}$$

where V_{fast} and V_{slow} are the fast and slow magnetosonic speeds, and V_{Ax} is the Alfven speed based on the x component of the magnetic field. These can be expressed as

$$V_{fast}^2 = \frac{1}{2}\left[c_s^2 + V_A^2 + \sqrt{\left(c_s^2 + V_A^2\right)^2 - 4c_s^2 V_{Ax}^2}\right], \tag{9}$$

$$V_{slow}^2 = \frac{1}{2}\left[c_s^2 + V_A^2 - \sqrt{\left(c_s^2 + V_A^2\right)^2 - 4c_s^2 V_{Ax}^2}\right], \tag{10}$$

$$V_{Ax}^2 = \frac{B_x^2}{\mu_o \rho}. \tag{11}$$

Here, c_s is the ion sound speed, which for a perfect gas is

$$c_s^2 = \frac{\gamma p}{\rho}. \tag{12}$$

Information propagates along characteristics which travel at wave speeds given by the eigenvalues. Most modern numerical techniques for solving hyperbolic equations are based upon the idea of splitting the fluxes into components due to left and right running waves. Then each part of the flux can be differenced in an upwind manner, which greatly reduces numerical oscillations and stabilizes the solutions.

It is well known that if a hyperbolic equation is solved with an explicit scheme, then the allowable time step to maintain numerical stability is given by the CFL (Courant-Friedrichs-Lewy) condition, which in the case of the 1D MHD equations is

$$\Delta t < \frac{\Delta x}{|v_x + V_{fast}|}. \tag{13}$$

For the high magnetic fields and low densities common in many plasma experiments, the fast magnetosonic speed is quite high, and thus the time step is prohibitively small. We are often interested in only modeling the physics that occurs slower than Alfvén time scales. For example, it can be shown that resistive tearing modes, which are important in studying fusion plasmas, evolve on a time scale that is given by[5]

$$\tau_{tearing} \propto \tau_A^{2/5} \tau_\eta^{3/5} = (Lu)^{3/5} \tau_A. \tag{14}$$

τ_A is the Alfvén time, τ_η is the resistive diffusion time, and Lu is the Lundquist number, which is given by

$$Lu = \frac{\tau_\eta}{\tau_A} = RmAl. \tag{15}$$

If Lu is 10^6, which is typical for laboratory plasmas in fusion applications, the resistive tearing time is approximately 4000 times larger than the Alfvén time. By treating the hyperbolic fluxes implicitly in time, the stability restriction on the time step is removed, and the solution can be advanced at the larger resistive tearing time step. This is our motivation for proposing an implicit scheme.

The starting point for deriving the algorithm is the two-dimensional ideal MHD equations in Cartesian form

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0. \tag{16}$$

We then discretize eqn(16) using first order Euler time differencing to get

$$\frac{\left(Q_{ij}^{n+1} - Q_{ij}^{n}\right)}{\Delta t} = -R_{ij}\left(Q^{n+1}\right) = -R_{ij}^{n+1} \tag{17}$$

where R is

$$R_{ij} = F_{i+1/2,j} - F_{i-1/2,j} + G_{i,j+1/2} - G_{i,j-1/2}. \tag{18}$$

Note that in this equation and all that follow the grid metric terms (cell areas and volumes) are omitted for clarity. We linearize R as follows:

$$R_{ij}^{n+1} \approx R_{ij}^{n} + \left(\frac{\partial R}{\partial Q}\right)_{ij}^{n} \left(Q_{ij}^{n+1} - Q_{ij}^{n}\right) \tag{19}$$

Substituting this expression back into eqn(17) and rearranging, we get

$$\left[\frac{\bar{\bar{\mathrm{I}}}}{\Delta t} + \left(\frac{\partial R}{\partial Q}\right)_{ij}^{n}\right] \Delta Q_{ij}^{n} = -R_{ij}^{n}. \tag{20}$$

Here ΔQ is defined as

$$\Delta Q^{n} \equiv Q_{ij}^{n+1} - Q_{ij}^{n}. \tag{21}$$

The left hand side of the eqn(20) is an implicit operator operating on ΔQ. It is a large banded block matrix. In three dimensions, it is an $(I_{max} \times J_{max} \times K_{max})$ by $(I_{max} \times J_{max} \times K_{max})$ matrix, where I_{max} is the number of cells in the x direction, etc. It is quite costly to invert a matrix of this size directly. We choose to invert it using an approximate factorization, which can be done more efficiently. When solved this way, eqn(20) is no longer time accurate. However, we can still achieve time accuracy with this type of scheme by adding the time derivative of Q as a source term to the right hand side of the equation. We then have

$$\left(\frac{\partial Q}{\partial \tau}\right)^{n+1} = -R_{ij}^{n+1} - S_{ij}^{n+1} \tag{22}$$

where

$$S_{ij}^{n+1} = \frac{1}{2\Delta t}\left(3Q_{ij}^{n+1} - 4Q_{ij}^{n} + Q_{ij}^{n-1}\right) \approx \frac{\partial Q}{\partial t}. \tag{23}$$

The τ in eqn(22) can be thought of as a pseudo time variable. At each physical time step, eqn(22) is solved iteratively in pseudo time until the left hand side vanishes. When the solution converges, our original equation

$$\frac{\partial Q}{\partial t} = -R \tag{24}$$

is solved. This technique is known as dual time-stepping.[6] Note that in eqn(23) a more accurate time derivative can be used at the expense of the additional memory needed to store the Q vectors from previous time steps.

One advantage of the strategy outlined above is that the implicit operator and the right hand side in eqn(20) are decoupled. The structure of the matrix no longer depends on the details of the discretization of the right hand side fluxes. In the following sections we will describe the relaxation scheme that is used to iteratively invert the implicit operator and the approximate Riemann solver that is used to form the right hand side fluxes.

2.2.3 LU-SGS Relaxation Scheme

We have used the lower-upper symmetric-Gauss-Seidel (LU-SGS) method to iteratively invert the implicit operator.[7] To derive this method, we first consider the following first order accurate flux-vector splitting of R (at time level $n+1$) in eqn(17):

$$R_{ij} = F_{ij}^{+} - F_{i-1,j}^{+} + F_{i+1,j}^{-} - F_{ij}^{-} + G_{ij}^{+} - G_{i,j-1}^{+} + G_{i,j+1}^{-} - G_{ij}^{-} \tag{25}$$

where F^{+} is the portion of the F flux vector corresponding to right-running waves, and F^{-} is the portion corresponding to left-running waves, and G^{+} and G^{-} are similarly defined. This equation is linearized to obtain

$$\left\{\bar{\bar{\mathrm{I}}} + \Delta t\left(A_{ij}^{+} - A_{i-1,j}^{+} + A_{i+1,j}^{-} - A_{i,j}^{-} + B_{ij}^{+} - B_{i,j-1}^{+} + B_{i,j+1}^{-} - B_{i,j}^{-}\right)\right\} \times \Delta Q_{ij}^{n} = -\Delta t R_{ij}^{n} \tag{26}$$

where A^{+} is the Jacobian of F^{+}, and so on. We approximate these Jacobians as

$$A^{+} = \frac{1}{2}\left(A + \rho_A\right) \tag{27}$$

$$A^- = \frac{1}{2}(A - \rho_A) \tag{28}$$

where ρ_A is the maximum eigenvalue (spectral radius) of A. If we then iteratively solve this simplified implicit operator using a forward Gauss-Seidel sweep followed by a backward sweep, the resulting algorithm can be written as

$$\begin{aligned}\left\{\bar{\bar{\mathrm{I}}} + \Delta t\left[(\rho_A + \rho_B)\bar{\bar{\mathrm{I}}} - A^+_{i-1,j} - B^+_{i,j-1}\right]\right\} \\ \times \left\{\bar{\bar{\mathrm{I}}} + \Delta t\left[(\rho_A + \rho_B)\bar{\bar{\mathrm{I}}} + A^-_{i+1,j} + B^-_{i,j+1}\right]\right\} \\ \times \Delta Q^n_{ij} = -\left[1 + \Delta t\,(\rho_A + \rho_B)\right]\Delta t R^n_{ij}\end{aligned} \tag{29}$$

The forward sweep is equivalent to inverting a lower block diagonal matrix [the first braced term in eqn(29)], and the backward sweep is equivalent to inverting an upper block diagonal matrix [second braced term in eqn(29)]. This structure leads naturally to several vectorization and parallelization strategies.

If we sweep through the computational domain along lines of constant $i + j$ (in 2-D), each term along these lines is independent of the others and depends only on data that has already been updated during the current sweep. This type of fine grain parallelization is ideal for vector computers. However, that degree of parallelism is not efficient for parallel computers because the extra communication time between processors exchanging data more than offsets the gain in computational efficiency. To optimize this algorithm for a parallel architecture, we need to break up the computational domain into blocks and send each block to a different processor. At the boundaries between the blocks, we reduce the data dependency between the blocks by using data from the previous time step along the block boundaries. This effectively reduces those points into a Jacobi iteration. However, the interior points are still solved with a Gauss-Seidel iteration. As long as the blocks are large enough that there are many more interior points than boundary points, then the overall convergence rate is approximately the same as Gauss-Seidel.

2.2.4 More Powerful Relaxation Schemes

As explained in the previous section we have successfully used the LU-SGS method to iteratively invert the implicit operator. The method is based on approximating the flux Jacobians as defined by eqns(27) and (28). We have found that the approximate natures of both the solver and the inversion method can lead to slow numerical convergence. We have begun to

investigate more powerful inversion methods. These methods use numerically calculated flux Jacobians and formulate the implicit equation in the standard matrix form:

$$\bar{\bar{A}}\mathbf{x} = \mathbf{b} \tag{30}$$

as in eqn(20). The implicit equation can then be solved using any of the more powerful iterative solvers, such as conjugate gradient or multigrid.

The flux Jacobians may be calculated numerically using an "epsilon" approach. At each position in the domain, the flux Jacobian is calculated using

$$\frac{\partial F}{\partial Q} = \frac{F(Q+\epsilon) - F(Q)}{\epsilon} \tag{31}$$

ϵ is a vector of length 8. Its value is chosen to be a small number but larger than machine zero.

An alternative and more intriguing method to numerically calculate the flux Jacobian is to use complex number space. We expand the flux in a Taylor series.

$$F(Q+ih) = F(Q) + ih\frac{\partial F}{\partial Q} - h^2\frac{\partial^2 F}{\partial Q^2} - ih^3\frac{\partial^3 F}{\partial Q^3} + ... \tag{32}$$

Solving for the flux Jacobian, we have

$$\frac{\partial F}{\partial Q} = \Im\left[\frac{F(Q+ih)}{h} + O(h^3)\right] \tag{33}$$

The flux Jacobian calculated in this manner is third order accurate and, therefore, does not require as small a value of h.

We will continue to investigate these more exact and powerful formulations of the implicit equation.

2.2.5 Approximate Riemann Solver

The fluxes on the right hand side of eqn(20) are discretized using a Roe-type approximate Riemann solver.[8] In this method the overall solution is built upon the solutions to the Riemann problem defined by the discontinuous jump in the solution between each pair of cells. The numerical flux for a first-order accurate (in space) Roe-type solver is written in symmetric form as

$$F_{i+1/2} = \frac{1}{2}\left(F_{i+1} + F_i\right) - \frac{1}{2}\sum_k l_k \left(Q_{j+1} - Q_j\right)|\lambda_k| r_k \tag{34}$$

where r_k is the k^{th} right eigenvector, λ_k is the absolute value of the k^{th} eigenvalue, and l_k is the k^{th} left eigenvector. The values at the cell interface $(i+1/2)$ are obtained by a simple average of the neighboring cells. These first order accurate upwind fluxes are used in the vicinity of sharp discontinuities in order to suppress oscillations in the solution. We achieve a globally second order accurate solution by using a flux limiter that modifies the first order flux so that it uses second order central differencing in smooth portions of the flow. We are using a minmod limiter.[9]

Once the eigenvalues and eigenvectors are obtained and properly normalized to avoid singularities, it is relatively straight-forward to apply this scheme to the one-dimensional ideal MHD equations.[10, 11] Unlike for the Euler equations, the extension to more than one dimension is non-trivial. The reason is that in more than one dimension, the Q vector must include B_x in addition to the other magnetic field components. (For the one-dimensional case B_x is constant by virtue of $\nabla \cdot \mathbf{B} = 0$). Since the $\mathbf{j} \times \mathbf{B}$ force acts perpendicularly to the directions of $\mathbf{j}$ and $\mathbf{B}$, the F flux vector has a zero term corresponding to B_x. Thus, the Jacobian matrix of F is singular and has a zero eigenvalue. This means we no longer have a complete set of physically meaningful eigenvectors. Physically, we expect information to travel either at the fluid velocity or at the fluid velocity plus or minus the wave speeds. Simply dropping B_x from the equation set is not a viable option, because B_x needs to change in order to maintain $\nabla \cdot \mathbf{B} = 0$. Powell *et al.*, recently solved this problem by modifying the Jacobian in such a way as to change the zero eigenvalue to v_x (keeping the others unchanged), and then adding in a source term that exactly canceled out the terms introduced by the modified Jacobian.[12]

The source term is

$$S_{div} = - \begin{bmatrix} \rho \\ \mathbf{B} \\ \mathbf{v} \\ \mathbf{v} \cdot \mathbf{B} \end{bmatrix} \nabla \cdot \mathbf{B} \tag{35}$$

It is proportional to the divergence of $\mathbf{B}$ and thus very small.

2.2.6 General Multiblock Implementation

General multiblock grid capability has been implemented in our code. This capability allows the application of the code to arbitrarily complex geometries which is one of the primary project objectives. MACH3 also uses a multiblock grid though not as general as the one implemented here. The

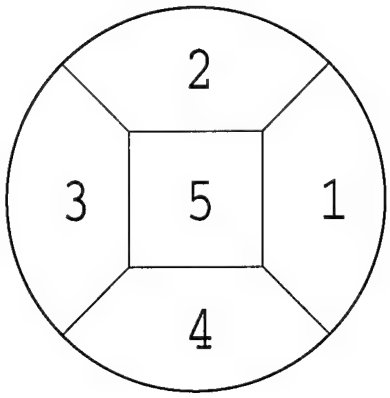


Figure 1: Top view of a non-simply connected grid. (Only the block boundaries are shown for clarity.)

grid that spans the physical domain of interest is composed of blocks which have a local grid. Blocks share faces with adjacent blocks thus allowing information to pass through the whole domain. The only restriction is that the cells on a block's face must map one to one to the cells on the adjacent block's face. An advantage of this approach is that the user is able to generate highly orthogonal grids with cell aspect ratios close to unity even for complex geometries. This approach is readily adaptable to parallelization by domain decomposition where the blocks are distributed among the processors of a parallel computer.

An advantage of multiblock grids is the flexibility they provide for complex geometries. This flexibility is even greater if there is no limitation to simply-connected blocks. See Figure 1 for an example of a non-simply connected block grid. This means that an arbitrary number of blocks may meet at a block a vertex. An example of this type of grid is illustrated in Section 3.7.

The MHD algorithm is now nested within a loop over grid blocks. Logic has been implemented in the new algorithm that allows the user to freely distribute the blocks on processors. A static load balancing algorithm can be implemented to distribute the blocks such that the number of cells on each processor is approximately equal. This strategy balances the load among the processors and makes the most efficient use of the parallel computer. The appropriate relationship between the blocks is maintained by a connectivity

matrix. Blocks sharing a face transfer data either by simply copying if they reside on the same processor or by message passing if they reside on different processors.

Multiblock implementation led to memory management issues that we solved by using Fortran 90. The most useful feature used was dynamic memory allocation. Dynamic memory allocation allows the specification of array sizes at run time and leads to a significant memory savings compared to static memory allocation, where a maximum overall array size has to be specified at compile time. Memory allocation issues are exacerbated in the multiblock algorithm where blocks can have widely varying numbers of cells. While most Fortran 77 compilers implement dynamic memory allocation, the implementations are not standard. Using Fortran 90 we maintained the portability of the code.

2.2.7 Second-Order Accurate Boundary Conditions

For simplicity an early version of the algorithm used first-order accurate boundary conditions. Practically this meant that a single layer of ghost cells were used for all boundaries. This arrangement was shown to lower the overall accuracy of the algorithm. We have since implemented second-order accurate boundary conditions by using two layers of ghost cells. The second-order accurate boundary conditions ensure that our simulations are completely independent of how the grid is decomposed.

Since the code is three dimensional the data passed between blocks is represented as a pair of two dimensional structures. Transfer of these two dimensional structures by message passing uses derived data types. The Message Passing Interface (MPI) provides a set of procedures for defining derived data types.[13] A derived data type is a template that describes how a complex structure is built from a more basic data type. In our approach the most basic unit of data is a cell. A cell contains the eight conserved variables mass density (ρ), components of momentum (ρv_x, ρv_y, ρv_z), components of magnetic field (B_x, B_y, B_z) and energy density (e). A cell can be thought of as a zero dimensional data structure. A strip is a one dimensional structure made of cells. A face is a two dimensional structure made of strips. Finally, a double face is made of two adjacent faces. The composition of the derived data type is illustrated in Figure 2.

Use of derived data types is more general and speeds the message transfer, compared to explicitly constructing the messages. Our results show a three to ten fold improvement in message passing time when derived data types are used.

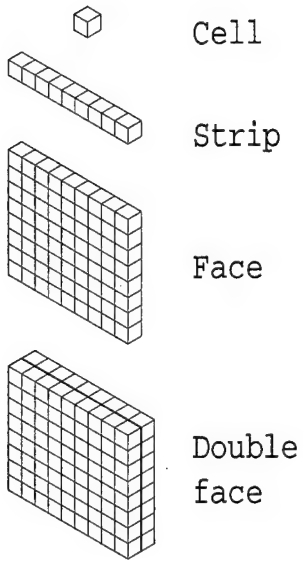


Figure 2: Hierarchy of the derived data types. The most basic derived data structure is the cell.

2.2.8 Unaligned Finite Volumes

The deficiency revealed by the spheromak simulations was corrected by recasting the algorithm using an unaligned finite volume formulation. Using the identical finite formulation for the hyperbolic fluxes as for the parabolic fluxes is essential to achieving our objective of a fully-coupled MHD code. The previous algorithm used a generalized coordinates approach to calculate the fluxes across cell interfaces.[14] These fluxes are used to update the solution at the next time step. The flux in generalized coordinates is

$$\mathcal{F}^{\iota} = F n_x^{\iota} + G n_y^{\iota} + H n_z^{\iota}, \tag{36}$$

where F, G and H are the fluxes in Cartesian coordinates, n_x, n_y and n_z are the components of the unit normal vector, and $\iota = 1, 2, 3$ stands for each of the three generalized coordinates (ξ, η, ζ). These generalized coordinates fluxes $\mathcal{F}^{\iota}$ are calculated in the cells to the left and right of the interface. The flux at the interface is obtained by averaging the two.

$$\mathcal{F}_{i+\frac{1}{2}} = \frac{1}{2}(\mathcal{F}_i + \mathcal{F}_{i+1}) - \frac{1}{2}\sum_{k=1}^{8} \alpha_k |\lambda_k| r_k, \tag{37}$$

where α_k are the wave strengths, λ_k are the wave speeds, and r_k are the right eigenvectors of the flux Jacobian. The eigenvectors and eigenvalues of the flux Jacobian $(\partial\mathcal{F}/\partial Q)$ are evaluated using average values of the conserved variables. Presently the code uses arithmetic averages.

In the new approach, the fluxes at an interface are calculated based on a locally aligned coordinate system. The Riemann problem is then solved at the interface in a natural direction to generate fluxes that are automatically orthogonal to the interface. A divergence theorem can now be applied to the finite volume cell to calculate the change in the conserved variables. See Figure 3 for an illustration of the local coordinate systems. The unaligned finite volume methods is exactly the same method that has already been implemented for the calculation of the parabolic fluxes. Using the same approach for the hyperbolic fluxes will make the code consistent.

We locally rotate the coordinate system such that one axis is normal to the interface. The Riemann problem is solved along the axis normal to the face. The rotation is kept consistent by performing a two step rotations about the Cartesian coordinates. We calculate the angles between each of the original axes (x^o, y^o, z^o) and the normal to the face. The axis corresponding to the minimum angle is aligned with the normal. For example, if x^o is to be aligned with the normal then the first rotation is about the

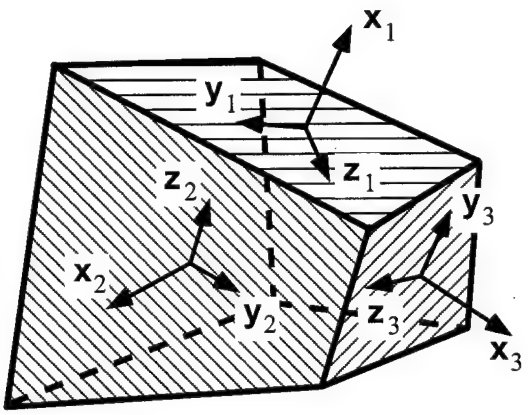


Figure 3: Schematic drawing of a finite volume cell with local coordinate systems at its interfaces. The fluxes are only needed along the $\mathbf{x_i}$ directions.

z^o axis with an angle θ_1. The second rotation is about y', with an angle θ_2, which aligns the coordinate system with the interface normal. The new coordinate axes are (x_1, y_1, z_1). For this case the angles are

$$\theta_1 = atan\frac{n_y}{n_x}, \qquad \theta_2 = -atan\frac{n_z}{\sqrt{n_x^2 + n_y^2}}, \tag{38}$$

and the rotation matrices are

$$R_1(\theta_1) = \begin{bmatrix} cos\theta_1 & -sin\theta_1 & 0 \\ sin\theta_1 & cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad R_2(\theta_2) = \begin{bmatrix} cos\theta_2 & 0 & sin\theta_2 \\ 0 & 1 & 0 \\ -sin\theta_2 & 0 & cos\theta_2 \end{bmatrix}. \tag{39}$$

The rotation is illustrated in Figure 4.

Once the components of the vector fields (**v** and **B**) are rotated the Riemann problem is solved and the flux along the normal to the face is calculated. Then the conserved variables are updated and the vector fields are rotated back to the original Cartesian coordinate system.

Since the rotation matrices are orthogonal their inverses are equal to their transposes ($R^{-1} \equiv R^T$) so that the extra work to be performed by the algorithm at each cell is minimal (27 floating point operations). This method is currently implemented and is being tested.

This new approximate Riemann solver contains several advantages over the previous one. The primary advantage is the ability to model plasmas

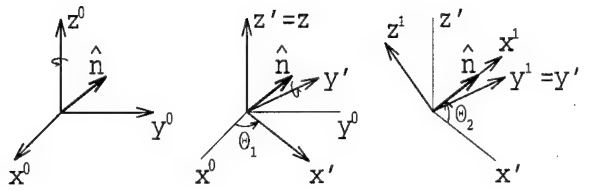


Figure 4: Rotation of a cell coordinate system that aligns x^o with the normal to the cell face $\hat{n}$.

in the low β regime. β is the ratio of thermal pressure to magnetic pressure. Many plasmas exist in the parameter space of high temperature but very low density and strong magnetic fields. The solver is now positively conservative according to the criteria described by Einfeldt.[15] This property extends the range of validity of the algorithm to the region where the magnetic and/or kinetic energies are large compared to the thermal energy. The improved accuracy can be seen in Figure 5 which shows a shock tube simulation with a pressure ratio of 100. The previous algorithm produces a remnant discontinuity at the original location of the shock.

Another advantage of the new solver is the ability to model strong rarefaction waves. Rarefaction waves which leave an almost perfect vacuum in their wake. Figure 6 shows the plot of a one dimensional simulation of double rarefaction waves using the new solver. The pressure in the wake of the rarefaction waves continues to decrease towards zero without ever becoming negative. The double rarefaction wave is a non-linearizable problem and is a rigorous test for an approximate Riemann solver. The flux formulation follows that of Roe[8] except the correction is performed by modifying the flux calculation at sonic points as opposed to a correction after the conserved variables have been advanced.

A modification was made to the algorithm as described by Aslan and Kammash.[16] The flux limiter which they investigated did not perform well in the situations where the Mach number was large, as it is in hypersonic flight simulations. We implemented a wave limiter[9] which performs well for a broader range of Mach numbers.

2.2.9 Parabolic MHD Terms

To this point we have only considered the hyperbolic terms. When finite viscosity and resistivity are included, the parabolic terms of the MHD equa-

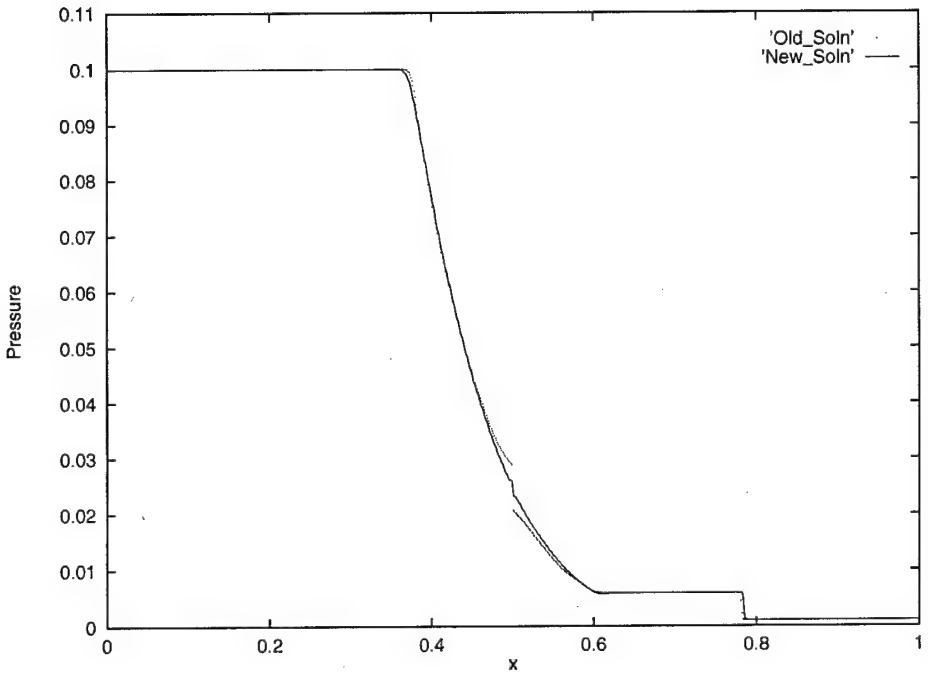


Figure 5: Plot of a one dimensional shock simulation (pressure) using the new solver and the old solver. Notice the remnant discontinuity shock is reduced by the new solver.

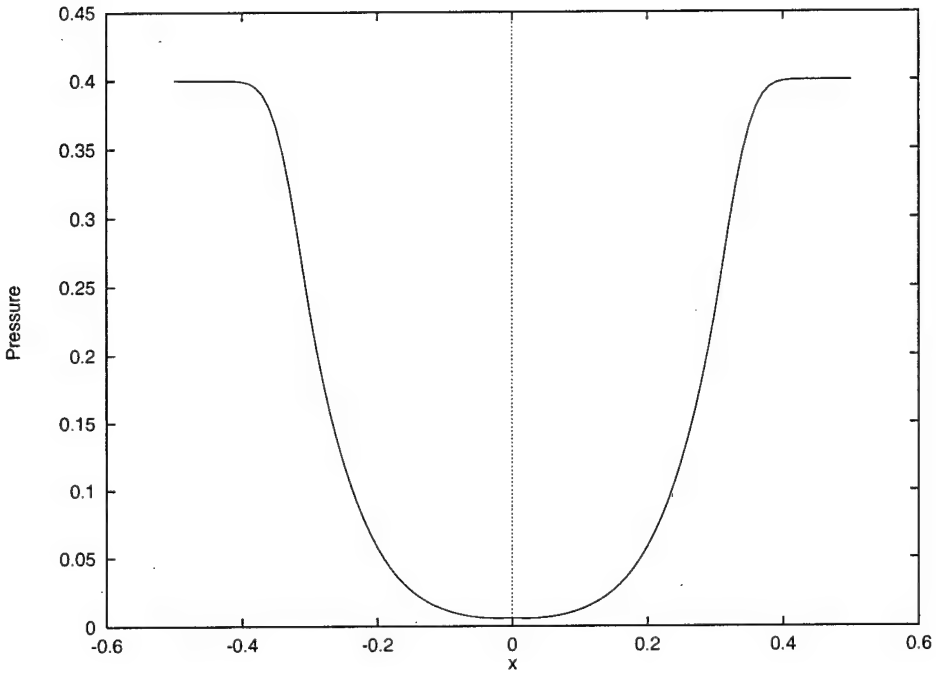


Figure 6: Plot of a one dimensional simulation of double rarefaction waves using the new solver. The pressure in the wake of the rarefaction waves continues to decrease towards zero without ever becoming negative.

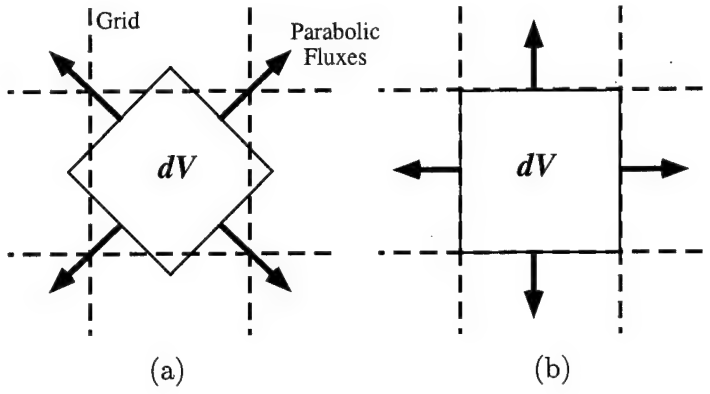


Figure 7: Positioning for (a) vertex-centered and (b) face-centered parabolic fluxes. The face-centered fluxes produced more accurate and stable solutions.

tions [right hand side of eqn(1)] become important. For reasonably large values of Re and Rm (easily in the range of interest for most applications), the parabolic terms can be differenced explicitly without constraining the allowable time step. In this work we difference the parabolic terms explicitly in time with central differences in space. They are added to the right hand side fluxes arising from the approximate Riemann solver.

During the past year, we have spent a concerted effort on the parabolic terms to achieve accurate and stable calculations. Originally we used the same flux centering scheme that is used in MACH3 where the fluxes are calculated at the cell vertices and a divergence law is applied around the cell center. See Figure 7(a). A detailed stability analysis demonstrates the potential for grid decoupling and a resulting odd-even instability. [This result has important implications to all ALE (arbitrary Lagrangian-Eulerian) codes and will soon be submitted to a journal.] Locating the parabolic fluxes at the cell faces which corresponds to the location of the hyperbolic fluxes produced solutions that converged faster and were more accurate than locating the parabolic fluxes at the cell vertices. See Figure 7(b).

We also point out that the resistive electric field term in eqn(1) is different than the one commonly used and presented last year $\nabla\cdot\bar{\bar{\eta}}\cdot\nabla\mathbf{B}$ which does not hold for spatially dependent anisotropic resistivity. Plasma resistivity is a strong function of temperature and of the orientation to the magnetic field. Therefore, the assumption of spatially constant isotropic resistivity is incorrect. The new term reduces from the conservative formulation of the

more general equation.

$$\int dV \nabla \times (\bar{\bar{\eta}} \cdot \nabla \times \mathbf{B}) = \oint dS \times (\bar{\bar{\eta}} \cdot \nabla \times \mathbf{B}) = \int dV \nabla \cdot \bar{\bar{\bar{\Xi}}} = \oint dS \cdot \bar{\bar{\bar{\Xi}}} \tag{40}$$

where the transverse resistive electric field tensor is defined as

$$\bar{\bar{\bar{\Xi}}} \equiv \begin{bmatrix} 0 & \eta_z (\partial_x B_y - \partial_y B_x) & \eta_y (\partial_x B_z - \partial_z B_x) \\ \eta_z (\partial_y B_x - \partial_x B_y) & 0 & \eta_x (\partial_y B_z - \partial_z B_y) \\ \eta_y (\partial_z B_x - \partial_x B_z) & \eta_x (\partial_z B_y - \partial_y B_z) & 0 \end{bmatrix} \tag{41}$$

The dimensionless numbers have been removed for clarity.

2.2.10 Hall Effect Physics

Hall effect physics has long been neglected by MHD simulation codes. The primary reason for this neglect is that Hall effect physics significantly complicates the solution algorithm and drives the time step to very small values. Hall effect physics is important for many applications, such as low density upper atmosphere plasmas, high power microwave production, and obviously Hall thrusters for space propulsion.

The Hall effect is embodied in the last term of the generalized Ohm's law given by

$$\eta \mathbf{j} = \mathbf{E} + \mathbf{v} \times \mathbf{B} - \frac{1}{ne} \mathbf{j} \times \mathbf{B}. \tag{42}$$

The Hall effect term represents the finite Larmor radius of the ions and the ability of ions and electrons to drift in different directions. The result of the Hall effect on the dispersion relation can be seen by linearizing the Hall MHD equations about a static equilibrium with a uniform magnetic field. The dispersion relation for a Cartesian geometry is

$$\frac{(\omega^2 - k_z^2)^2 - \frac{\omega^2}{\omega_{ci}^2} k_z^4}{\omega^2 - k_z^2 + \frac{\omega^2}{\omega_{ci}^2} k_z^2} = \frac{\pi}{2} \tag{43}$$

where ω_{ci} is the normalized ion cyclotron frequency and its inverse is proportional to the Hall parameter, $\omega_{ce}\tau_e$. The dispersion relation is plotted in Figure 8 for three values of the Hall parameter. A Hall parameter of zero corresponds to ideal MHD. The characteristic wave speeds of a system is given by ω/k. For ideal MHD the speed is simply the Alfvén speed. When

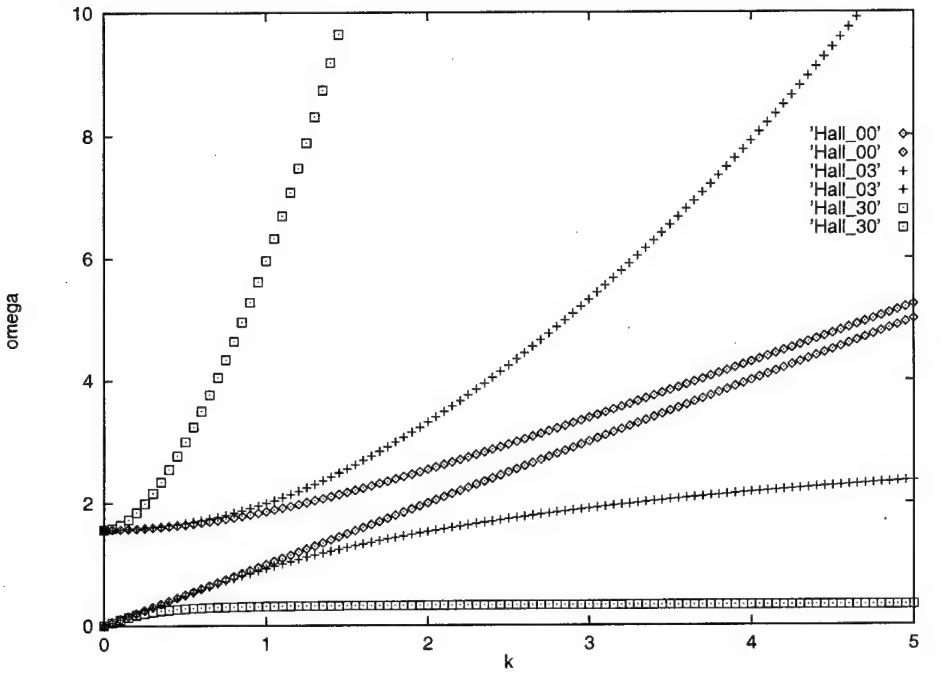


Figure 8: Dispersion relation for three values of the Hall parameter, $\omega_{ce}\tau_e$. A Hall parameter of zero corresponds to ideal MHD.

the Hall effect is included, the characteristic wave speed increases sharply as evident from Figure 8. The large wave speeds are difficult to stabilize and drive the simulation time step to very small values.

We are developing a semi-implicit method to treat the Hall effect terms in a manner which allows arbitrarily large time steps without any destabilizing behavior. The method is based on one that has been successfully installed in pseudo-spectral codes.[17] The equations which include the Hall effect terms contain a fourth-order dissipation component which is used to smooth and stabilize the numerical errors that are generated by taking large time steps. In a simplified form, the semi-implicit method updates the magnetic field according to

$$\begin{aligned} &\mathbf{B}^{n+1} + \Delta t^2 \left(\mathbf{C}_H \cdot \nabla\right)^2 \nabla^2 \mathbf{B}^{n+1} = \\ &\mathbf{B}^{*} + \Delta t^2 \left(\mathbf{C}_H \cdot \nabla\right)^2 \nabla^2 \mathbf{B}^{*} - \frac{\Delta t}{\omega_{ci}\tau_A \rho} \nabla \times \left(\mathbf{j} \times \mathbf{B}\right)^{n+\frac{1}{2}} . \end{aligned} \tag{44}$$

Hall effect physics can lead to a drifting of otherwise static magnetic fields. Figure 9 shows the computational results for a plasma bounded in the horizontal direction and periodic in the other direction. The upper series of plots shows the motion of the magnetic field due to the Hall effect. ($k = 5.0$ and $\omega_{ci}\tau_A = 3.5$.) The lower series is for the same situation except the Hall effect physics is not used. The behavior is consistent with analytical solution for this simulation.

3 Benchmarks and Applications

3.1 Ideal MHD Test Problems

3.1.1 One-Dimensional Coplanar MHD Riemann Problem

This test problem served to validate the approximate Riemann solver, because the computed solution could be checked against the exact analytical solution. For the one-dimensional ideal MHD equations (variations in x only), the equation for B_x reduces to B_x is constant and drops from the equation set, eliminating the zero eigenvalue in this case. The coplanar MHD equations are obtained from the full one-dimensional ideal MHD equations by setting B_z and v_z to zero, thus allowing only planar flow and fields. This eliminates the $v_x \pm V_{Ax}$ eigenvalues, leaving a system of five equations with five eigenvalues. Mathematically, the Riemann problem is an initial boundary value problem in which there is initially a discontinuity in the data such that the left half of the domain is at one state and the right half of the domain is at another state. As the solution evolves in time, shock waves

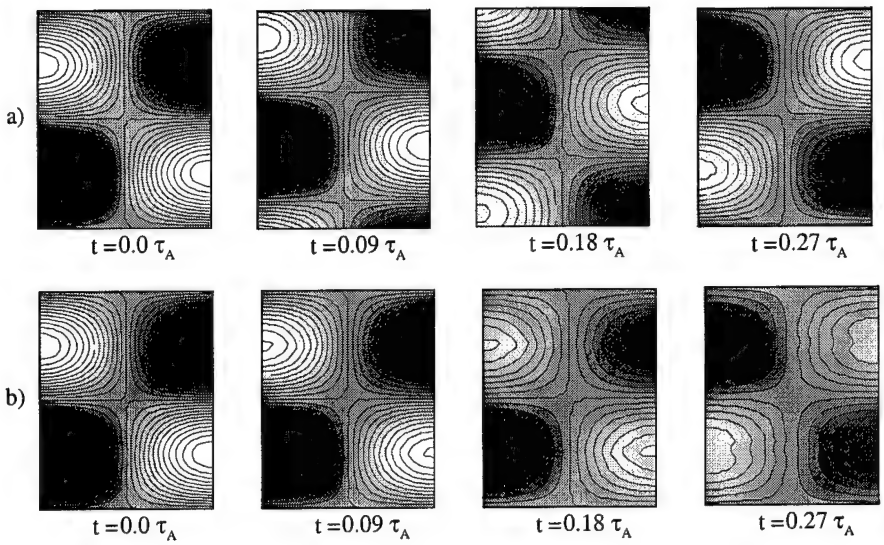


Figure 9: Hall effect induced motion of the magnetic field, $k = 5.0\ \omega_{ci}\tau_A = 3.5$. The lower series of plots is a simulation without the Hall effect. Shown are contours of out of plane magnetic field intensity.

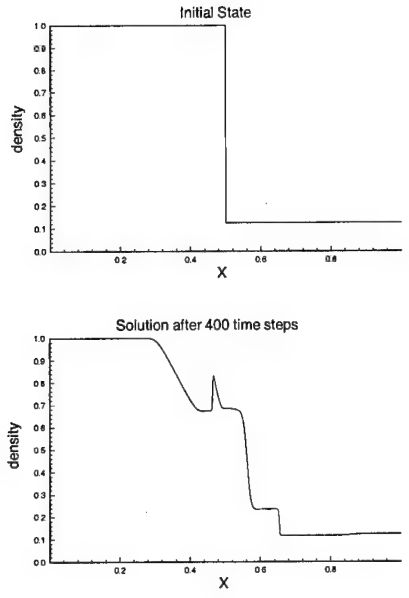


Figure 10: Numerical solution of coplanar Riemann problem. Density profile is shown initially and after solution has evolved for 400 time steps.

and rarefaction waves form and travel at speeds related to the wave speeds of the system. Although not physically realizable in plasmas, this problem is analogous to a shock tube in hydrodynamics.

For the full five-wave case, there is not a closed form analytical solution. Instead, the solution must be checked by calculating generalized Riemann invariants across the rarefaction waves and Rankine-Hugoniot jump conditions across the shock waves. Since this has already been done by Brio and Wu[10] for a specific set of conditions, for our test case we used the same initial conditions as they used in order to allow direct comparison with their published solution. The initial left state was $p = 1$, $\rho = 1$, and $B_y = 1$. The initial right state was $p = 0.1$, $\rho = 0.125$, and $B_y = -1$. The velocities were zero and B_x was 0.75. Figure 10 shows the initial density distribution and its numerical solution after 400 time steps on an 800 point grid with a CFL number of 0.8. Figure 11 is the corresponding plot of the transverse magnetic field (B_y). The solution was computed using explicit time-stepping. The solution clearly shows five waves formed corresponding to the five eigenvalues. They are a fast rarefaction wave, a slow shock, a contact surface moving to the right, a slow compound wave (rarefaction and

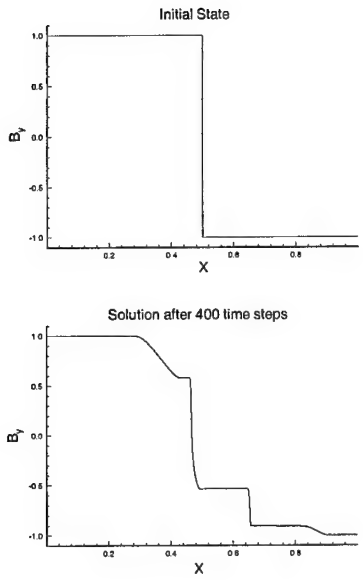


Figure 11: Numerical solution of coplanar Riemann problem. Transverse magnetic field profile is shown initially and after solution has evolved for 400 time steps.

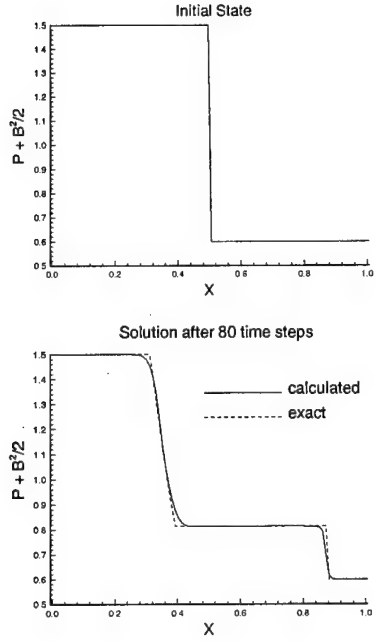


Figure 12: Comparison of numerical and exact solution of coplanar Riemann problem for $B_x = 0$ case.

shock), and a fast rarefaction wave moving to the left. Note that the numerical method is able to resolve the shocks over a few grid points without introducing numerical oscillations. This is one of the advantages of the flux splitting approach we have used. The computed solution overlaid exactly on Brio and Wu's published solution.

If we set $B_x = 0$ above, then the problem reduces to a hydrodynamic shock tube problem if one replaces the thermodynamic pressure by the sum of the thermodynamic and magnetic pressures. For this case one can find a closed form exact solution to compare to the calculated solution. Figure 12 shows both the calculated and the exact solution for $p + B^2/2$ after 80 time steps on a 100 point grid. There is very good agreement with the plateau values and the shock is resolved in a few cells without numerical oscillations.

3.1.2 Oblique Shock

This steady-state problem served primarily as a test of the LU-SGS implicit relaxation scheme. It also allowed us to examine the divergence of $\mathbf{B}$ at each

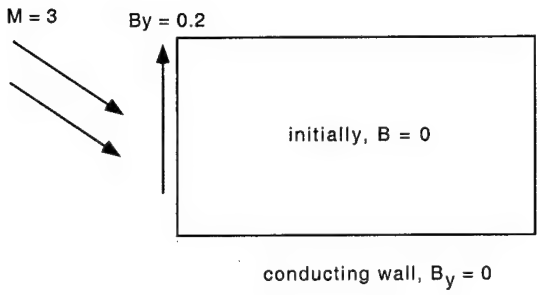


Figure 13: Geometry of oblique shock test problem.

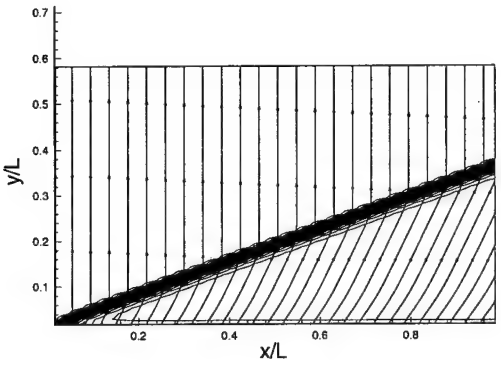


Figure 14: Density contours and field lines for an $M = 3$ flow impinging on a perfectly conducting plate at an angle of 25 degrees.

point to ensure that the zero eigenvalue fix was correctly implemented.

The geometry for these tests is shown in Figure 13. A super-Alfvénic flow (Mach number of 3) impinges on a perfectly conducting plate at an angle of 25 degrees. In addition, a vertical field of $B_y = 0.2$ is imposed at the left boundary. Since the plate is perfectly conducting, the component of the magnetic field normal to the plate is held at zero.

Figure 14 shows the steady-state solution of this problem. Contours of density and magnetic field lines are plotted. The density contours show that an oblique shock forms, as expected. Outside of the shock, the field is convected in from the boundary. At the shock, the field lines bend due to the change in direction of the flow at the shock. Finally, the field lines

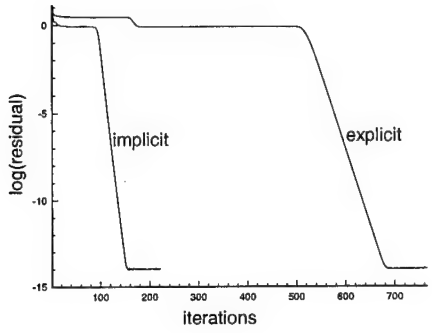


Figure 15: Logarithm of the two-norm of the energy equation residual plotted as a function of iteration number for explicit and implicit solutions of channel flow with horizontal velocity and vertical magnetic field imposed at the left boundary.

bend at the conducting wall as all the field is converted to B_x to satisfy the boundary condition while keeping the divergence of $\mathbf{B}$ equal to zero. We verified that the divergence was less than 10^{-14} throughout the domain.

This two-dimensional steady-state solution was obtained with explicit time stepping at a CFL number of 0.8 and with the LU-SGS implicit relaxation scheme at an infinite CFL number (approximate Newton iteration). Figure 15 is a plot of the logarithm of the two-norm of the residual of the energy equation as a function of the number of iterations. The implicit scheme converged to 10^{-14} in about 150 iterations, whereas the explicit scheme required about 700 iterations. This is an acceleration factor of about 4.5 for the implicit scheme. Higher acceleration factors can be achieved for finer grids.

3.2 Viscous and Resistive MHD Test Problems

The viscous and resistive terms in the MHD equations comprise the right hand side of the equality in eqn(1). The addition of these terms to the algorithm involved the modification of the R vector in eqn(20).

$$-R \rightarrow -R + \nabla \cdot \bar{\bar{T}}_p \tag{45}$$

The R vector is updated with each iteration to produce a solution that is fully coupled.

Using the divergence form of the parabolic terms reduces the differencing errors of the method. To preserve the accuracy on irregular meshes the derivatives are computed using a finite volume method.

The validation of the parabolic terms consisted of applying the code to a suite of test problems with known analytical solutions. We validated independently the terms associated with viscosity and those associated with resistivity and then the combined effect of all of the terms. The test problems were: (1) fully developed laminar flow between two parallel plates, (2) magnetic field generated by a constant current density, and (3) Hartmann flow. All of these test problems were run until a steady-state solution developed. The capability of the code to capture time-dependent physical effects was also tested by modeling the exponential resistive decay of the magnetic field generated in test problem 2.

3.2.1 Laminar Flow

We benchmarked the code to two types of laminar flows between infinite parallel plates. The plates restrict the steady-state flow to be one-dimensional. No magnetic fields are present. This reduces the MHD equations to the Navier-Stokes equations. In these simulations a no-slip boundary condition was applied to the fluid in contact with the plates.

The first type of flow to which we benchmarked was viscous flow generated by one plate moving relative to the other plate. With no pressure gradient, constant viscosity, and incompressible flow, the equations reduce to

$$(Re)^{-1} \nabla \cdot \bar{\bar{\tau}} = (Re)^{-1} \nabla^2 v_x = 0 \tag{46}$$

which is Laplace's equation. For finite viscosity (Re) the analytical solution for the flow velocity is

$$v_x(y) = V_0 \left(1 - \frac{y}{L}\right) + V_L \frac{y}{L} \tag{47}$$

where V_0 is the velocity of the plate at $y = 0$ and V_L is the velocity of the plate at $y = L$.

The errors between the analytical solution and the code generated solution were below 10^{-9} (the two-norm of the error between the solutions). We performed the same simulation with no viscosity ($Re \rightarrow \infty$). As would be expected, the flow velocity vanished everywhere except on the plates. When the viscous heating was modeled, a transient pressure gradient $p(y)$

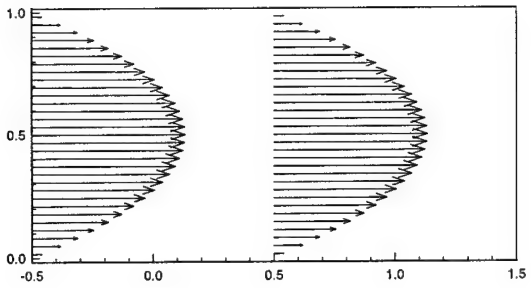


Figure 16: Simulation of laminar flow between parallel plates in the presence of a constant pressure gradient. The velocity profile is parabolic as expected from the analytical solution.

and transverse velocity $v_y(y)$ developed which heated the flow and increased its energy.

The next test was laminar flow between stationary parallel plates with a constant pressure gradient in the flow direction. The governing equation is

$$\nabla_x \cdot \left(p\bar{\bar{\mathrm{I}}}\right) = \frac{\partial p}{\partial x} = (Re)^{-1} \nabla^2 v_x. \tag{48}$$

The solution for this flow is the parabolic equation given by

$$v_x(y) = (Re) \frac{\partial p}{\partial x} \left(\frac{y}{L}\right) \left(\frac{L-y}{2L}\right). \tag{49}$$

Figure 16 shows the solution generated by the code. Again the errors were reduced to below 10^{-9}.

3.2.2 Resistive Diffusion

We benchmarked the resistive diffusion to a current sheet with a uniform current density. Values of the tangential magnetic field were specified at parallel infinite plates, in a similar way as the first of the laminar flow simulations.

For no flow velocity and constant resistivity the MHD equations reduce to a Laplace equation similar to eqn(46).

$$(Rm)^{-1} \nabla \cdot \nabla \mathbf{B} = 0 \tag{50}$$

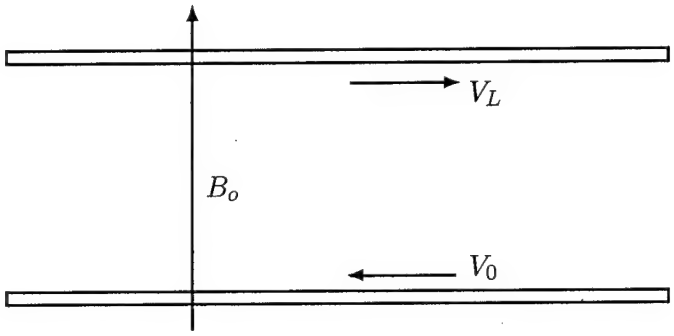


Figure 17: The Hartmann flow geometry showing the moving parallel plates and the cross magnetic field.

This equation has the same form for its solution as eqn(47).

$$B_x(y) = B_0\left(1 - \frac{y}{L}\right) + B_L\frac{y}{L} \tag{51}$$

where B_0 is the velocity of the plate at $y = 0$ and B_L is the velocity of the plate at $y = L$.

The code agreed with the analytical solution to within errors of 10^{-9}.

The time-dependent resistive decay of a magnetic field can be represented analytically by solving the one-dimensional transverse magnetic induction equation with constant resistivity.

$$\frac{\partial B_\perp}{\partial t} = (Rm)^{-1}\frac{\partial^2 B_\perp}{\partial x^2} \tag{52}$$

The solution is the exponential decay of the magnetic field with a sinusoidal profile.

$$B_\perp(t, x) \propto \exp\left(-\frac{\pi^2 t}{Rm}\right)\sin(\pi x) \tag{53}$$

for zero field boundary conditions at $x = 0$ and $x = 1$.

This simulation was performed beginning with a uniform field profile. The field decayed into the expected sinusoidal shape and the decay constant agreed with the analytical result to within 0.01%. The same test was repeated on a parabolic clustered grid with $\Delta x_{max}/\Delta x_{min} = 10$. The same accuracy was achieved.

3.2.3 Hartmann Flow

Hartmann flow combines the effects of viscosity and resistivity. The problem geometry is the same as that for the laminar flow with the addition of a magnetic field that is normal to the plates, in the y direction. See Figure 17 for an illustration.

The governing equations for the Hartmann flow can be found by combining the magnetic field and momentum equations from the MHD model. As before there will only be flow in the x direction. However, an applied electric field in the z direction must be included since it can generate an $\mathbf{E_o} \times \mathbf{B_o}$ flow in the x direction. The Hartmann flow is described by the differential equation,

$$\frac{\partial^2 v_x}{\partial y^2} - \frac{H^2}{L^2}\left(v_x + \frac{E_o}{B_o}\right) = 0, \tag{54}$$

where the Hartmann number is defined as

$$H \equiv \frac{B_o L}{\sqrt{\rho\nu\eta}} = AlL\sqrt{ReRm}. \tag{55}$$

The analytical solution to the Hartmann flow is

$$\begin{aligned} v_x(y) = V_0 \frac{\sinh\left(H(1-y/L)\right)}{\sinh(H)} + V_L \frac{\sinh(Hy/L)}{\sinh(H)} \\ -\frac{E_o}{B_o}\left[1 - \frac{\sinh\left(H(1-y/L)\right) + \sinh(Hy/L)}{\sinh(H)}\right] \end{aligned} \tag{56}$$

where the same no-slip boundary conditions have been applied. In the limit of no magnetic field, the solution reduces to the laminar flow solution, eqn(47).

The response of the magnetic field can be determined by solving the magnetic field equation for the field component that will be "dragged" with the flow. This magnetic field is described by

$$\frac{\partial B_x}{\partial y} = -\left(Rm\right)\left(v_x + \frac{E_o}{B_o}\right). \tag{57}$$

Using the flow solution of eqn(56), the solution for B_x is

$$B_x(y) = \left(\frac{Rm}{H}\right)\left(\frac{V_L - V_0}{2}\right)\left[\frac{\cosh(H/2) - \cosh\left(H(L-2y)/2L\right)}{\sinh(H/2)}\right]. \tag{58}$$

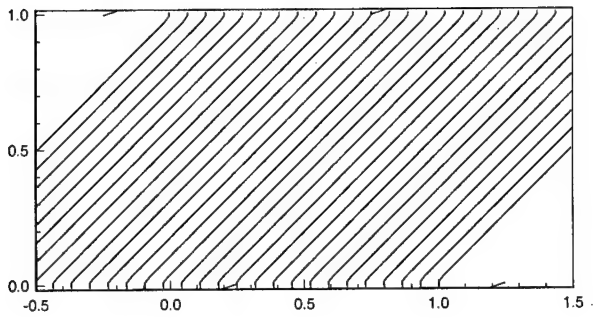


Figure 18: Hartmann flow simulation with $H = 10^4$. Flow velocity vectors and magnetic field lines are shown. The velocity of the flow is zero everywhere except at the plates. The magnetic field lines have a constant slope through the domain.

The boundary conditions are that B_x vanishes at the plates and the net current is zero. The first boundary condition may seem arbitrary, but it is consistent with the no-slip boundary condition applied to the flow solution. The second boundary condition relates the applied electric field, E_o, and the plate velocities.

$$\frac{E_o}{B_o} = -\frac{V_0 + V_L}{2} \tag{59}$$

Since the MHD equation set does not allow for an applied electric field, V_0 is set to $-V_L$, so that $E_o = 0$.

We performed simulations for large, small, and intermediate Hartmann numbers, H.

For a large Hartmann number, the effects of viscosity and resistivity are small, and the solution approaches that of ideal MHD. The flow velocity vanishes everywhere except on the plates, like it does for the inviscid case ($Re \rightarrow \infty$). The magnetic field is frozen into the plates and develops a slope (constant B_x) as the plates move. The slope of the magnetic field is determined by the value of H (the field lines slip through the plates due to resistivity). The slope of the magnetic field lines (B_o/B_x) is constant at H/Rm. Figure 18 shows the results from simulation with $H = 10^4$. A finite value of the flow velocity exists only at the plates. The magnetic field

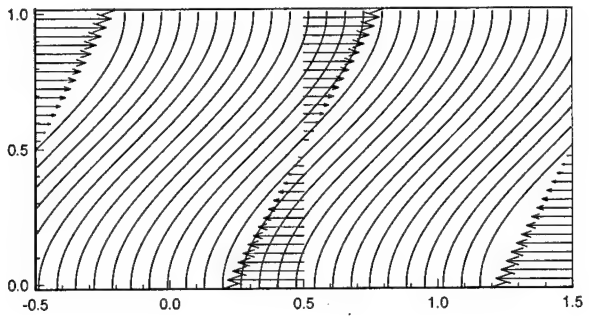


Figure 19: Hartmann flow simulation with $H = 0.1$. Flow velocity vectors and magnetic field lines are shown. The velocity profile is linear and the magnetic field lines have an "S" shape caused by the bulk fluid flow.

lines are straight except at the plates where B_x is forced to vanish because of the boundary conditions. For clarity the slope of the magnetic field has been normalized to unity at the midplane between the plates for all of the Hartmann flow simulations.

The limiting case of small Hartmann number is characterized by a flow that is dominated by viscous effects and a magnetic field that responds to the bulk fluid flow and the large resistivity. The flow velocity varies linearly from the velocity of the top plate to the velocity of the bottom plate, as described by eqn(47). The magnetic field diffuses through the plate and the bulk fluid, but the fluid drags the field lines along with the flow. This produces a swayed "S" shape to the field lines with a peak magnetic field at the midplane. Since the slope of the field lines is inversely proportional to the magnitude of B_x, the peak in the magnetic field corresponds to the field lines with the minimum slope (most horizontal). The minimum slope is $4/Rm$. The simulation results for $H = 0.1$ are shown in Figure 19. Notice the linear velocity profile and the swayed magnetic field lines.

Flows with Hartmann numbers in the intermediate ranges have solutions which exhibit characteristics of both of the limiting cases. The flow velocity falls to zero away from the boundaries in a scale length of L/H. This scale length is an appreciable fraction of the domain. The magnetic field is influenced by the motion of the plates and the fluid flow. The magnetic field has a swayed shape close to the plates and is linear around the midplane.

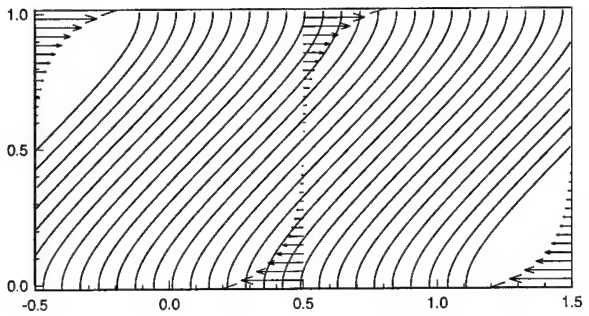


Figure 20: Hartmann flow simulation with $H = 10$. Flow velocity vectors and magnetic field lines are shown. The flow velocity only exists close to the plates. The magnetic field lines are linear around the midplane.

Away from the boundaries ($L/H < y < L - L/H$), the value of B_x is constant at $B_o Rm/H$. Figure 20 shows the results from a simulation with $H = 10$. The velocity profile falls to zero around the midplane. The magnetic field lines have a swayed shape like those in Figure 19 but not as dramatic, and they are linear around the midplane.

All of the Hartmann flow simulations converged to the analytical solution to within errors of 10^{-6}.

3.3 MPD Plasma Thruster

The magnetoplasmadynamic (MPD) thruster is an electric propulsion device for spacecraft. Electrical propulsion is a technological field that is important to the Air Force and industry for satellite station keeping and orbital maneuvering. This problem demonstrates the dual time-stepping algorithm, which allows flexible choice of time steps so that fast and slow transients can be tracked accurately and efficiently. This is also the first problem that exercises all of the parts of the new algorithm (the approximate Riemann solver, the LU-SGS relaxation scheme, the resistive and viscous terms, and the dual time-stepping) simultaneously. The problem geometry is shown in Figure 21. A current is applied across the left boundary. This current creates a magnetic field in the z direction that in turn leads to a $\mathbf{j} \times \mathbf{B}$ force that accelerates the plasma to the right. We expect that the plasma initially in the domain will be accelerated up to some exit velocity on a fast time

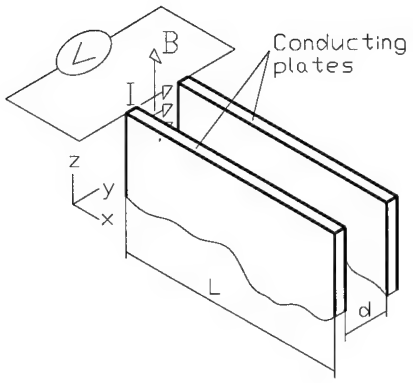


Figure 21: Geometry of the two-dimensional MPD thruster.

scale related to the Alfvén time. However, if there is a finite resistivity in the plasma, the magnetic field and current at the left boundary will diffuse into the domain on a slower time scale related to the resistive diffusion time. Ideally, one would like to take small time steps initially to follow the fast transient, and then switch to a much larger time step when the system is evolving more slowly.

If there is no viscosity, then the problem becomes one-dimensional in x, which is to the right in Figure 21. For this problem we chose a Lundquist number of 100, a reference magnetic field of 1 Tesla, a reference density of 10^{-5} kg/m^3, a reference length of 10 cm, and an imposed current of 30 kA. Figure 22 shows the plasma velocity as a function of x at several different times (normalized to the Alfvén time). The top plot shows the results of an explicit time-differencing simulation with a CFL number of 1. This simulation took 2600 time steps to advance the solution to $t = 10.17$. Notice that between 3 and 5 Alfvén times, the velocity reaches a constant uniform value along the length of the domain. The bottom plot is a simulation in which a CFL number of 1 was used until $t = 1.5$, at which point the CFL number was increased to 100 and the dual time-stepping implicit method was used to maintain stability. At each physical time step it took about 30 pseudo-time steps to converge, so the overall number of iterations was reduced to 1090 for the dual time-stepping case. The plots look similar to the explicit time-stepping results, except that the end of the fast transient is filtered out by taking such large time steps. On the other hand, Figure 23 shows that the magnetic field, which evolves on the

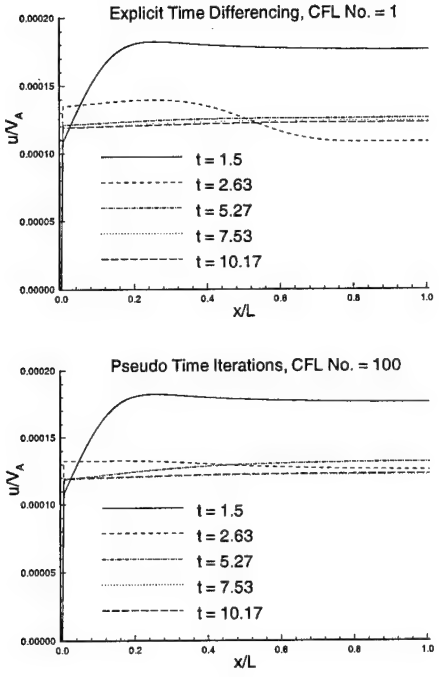


Figure 22: Plasma velocity as a function of x and time for explicit time differencing simulation and dual time-stepping (implicit) simulation.

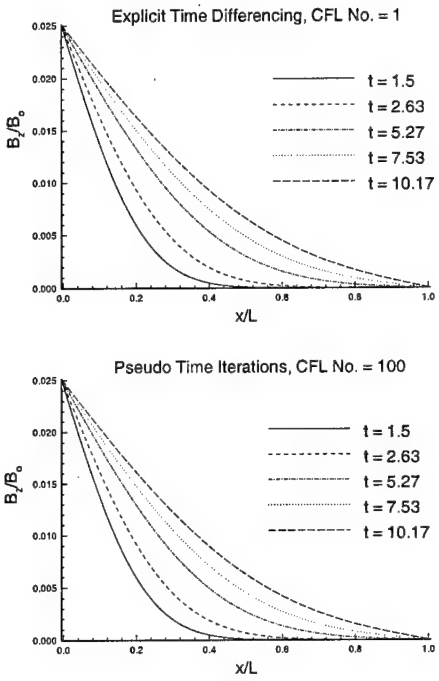


Figure 23: Magnetic field as a function of x and time for explicit time differencing simulation and dual time-stepping (implicit) simulation.

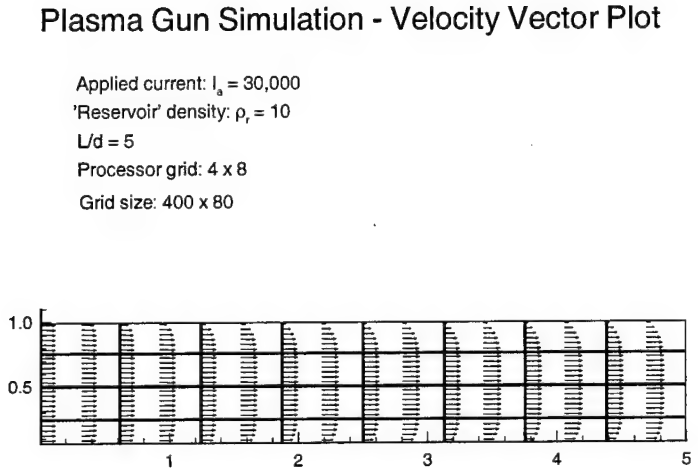


Figure 24: Velocity vectors for the MPD thruster. The plasma is accelerated down the gun by the $\mathbf{I} \times \mathbf{B}$ force and a boundary layer develops. The internal blocks illustrate the decomposition of the domain used for the validation of the parallel version of the code.

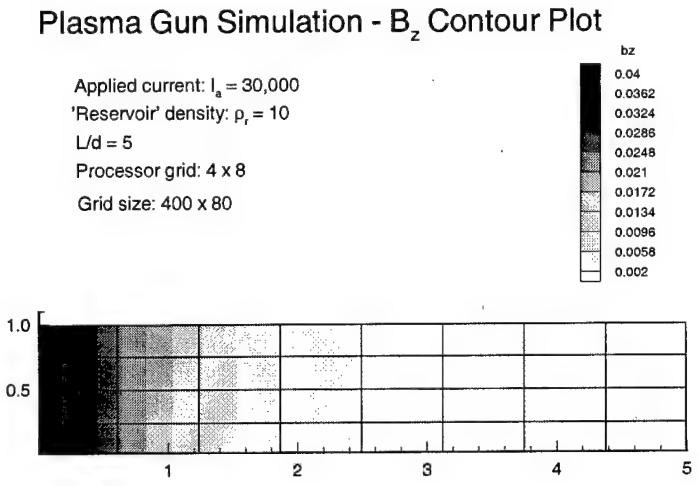


Figure 25: Magnetic field (B_z) contours for the MPD thruster. The gradient in the magnetic field produces the force applied to the plasma.

slower resistive diffusion time scale, is captured equally well by the explicit and implicit schemes. The development of the plasma velocity and internal magnetic field can be seen in Figures 24 and 25.

We have studied the magnetoplasmadynamic (MPD) plasma thrusters. Our goal is to validate the code against computational and experimental results of Sleziona *et al.*[18] and to improve the thruster design.

We have studied channel MPD thrusters successfully during tests of our code. The current work modeled the more realistic annular MPD thrusters. In an annular MPD thruster a current is driven through the plasma radially by coaxial electrodes (the anode is at the larger radius). The self-generated magnetic field and the current give the $\mathbf{j} \times \mathbf{B}$ force that accelerate the plasma.

A Hall plasma thruster also has a coaxial configuration. A radial magnetic field and an axial electric field are applied which produces an azimuthal current by the Hall Effect. The Hall current is created by electron drifting azimuthally at a high speed. The electrons ionize the injected gas propellant to form a plasma. Then the Hall current interacts with the radial magnetic field to accelerate the plasma axially.

Our simulation uses a grid composed of four blocks (see Figure 26) in an axisymmetric configuration. The electrodes are modeled as perfect conductors. The inlet gas was injected at a pressure that is 5% higher than the initial pressure inside the thruster. The expansion region beyond the exit of the thruster is also modeled to examine the plume of the thruster. The thruster current was held constant at $1kA$. The plasma temperature was $0.5keV$.

Results are encouraging and further investigation is required in order to validate the code against the mentioned experimental results. In Figure 27 we present the velocity vectors and contours of magnetic field after approximately 20 Alfvén transit times. The radial dependence of the velocity is caused by the radial dependence of the $\mathbf{j} \times \mathbf{B}$ accelerating force. A vortex ring is observed shedding from the cathode, in the upper quarter of the domain. The magnetic field balloons beyond the annular region between the electrodes as measured in experiments.

3.4 Magnetic Reconnection

In this application we present results demonstrating agreement between theoretical linear growth rates of the resistive instability in a sheet pinch and our non-linear resistive MHD code. We study resistive instabilities because they are a likely candidate for driving magnetic relaxation in the Helicity Injected Tokamak (HIT). The planar sheet pinch is a well understood

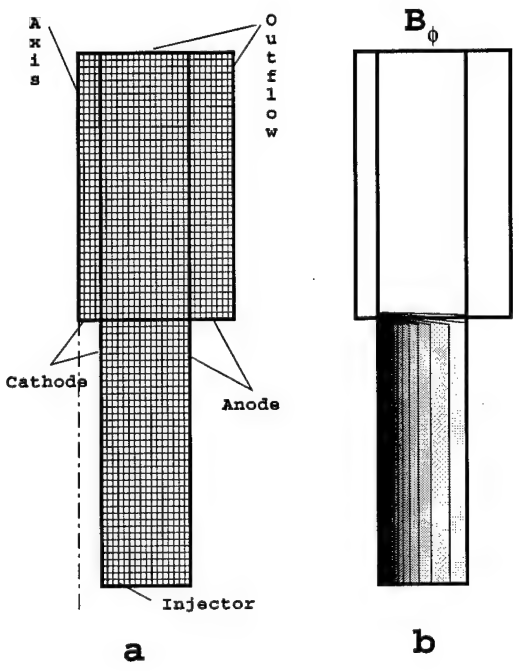


Figure 26: (a) Four block grid for the axisymmetric MPD simulation and (b) the initial contours of the magnetic field. (Black contours represent higher values.)

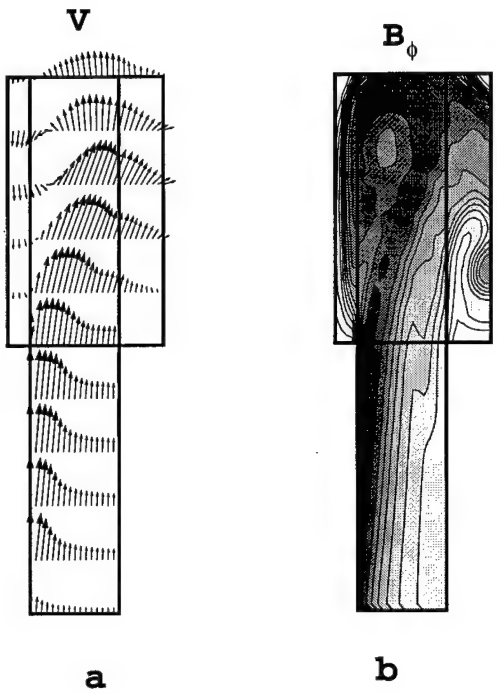


Figure 27: (a) Flow field and (b) contours of azimuthal magnetic field. (Black contours represent higher values.) Note the vortex ring shedding off the cathode.

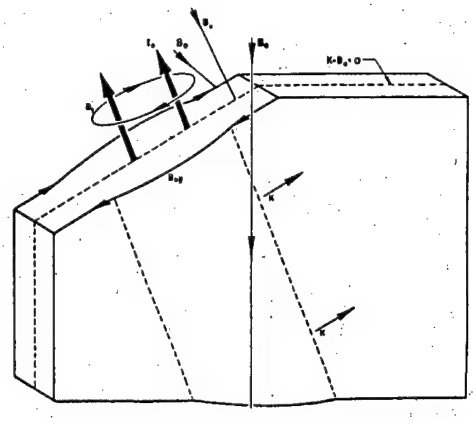


Figure 28: Schematic of planar sheet pinch problem [from H. P. Furth, Phys. Fluids **28**(6), 1595 (1985)].

configuration[19, 20, 21, 22] and provides a good test problem and benchmark for our MHD code.

We present the linear analysis of the sheet pinch.[20, 21] The linear equations are solved numerically to obtain the eigenmodes. The eigenvalues (growth rates) are compared with the analytical theory.[19] We then present the nonlinear analysis where our implicit MHD code is applied. A perturbation is initialized in the MHD code. The instability resulting from the perturbation is allowed to develop and finally saturates due to non-linear effects. The initially linear growth rate agrees with linear analysis.

3.4.1 Problem Description

We study the resistive instability in a planar sheet pinch, the symmetric tearing mode in a finite-thickness current sheet. See Figure 28 for schematic representation. For simplicity we examine the mode with the wave vector parallel to the equilibrium magnetic field.

$$k \parallel B_o \tag{60}$$

We define

$$F \equiv \frac{\mathbf{k} \cdot \mathbf{B}}{\mathbf{B_{ref}}} = \tanh\left(\frac{y}{a}\right) \tag{61}$$

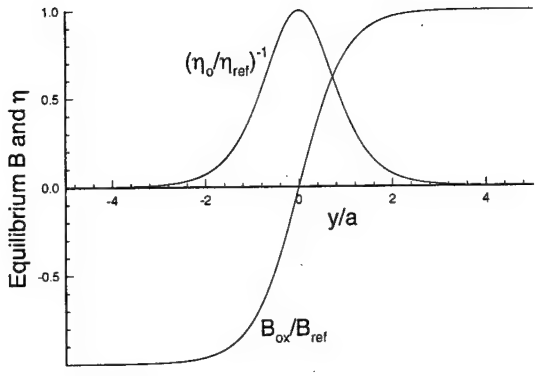


Figure 29: Equilibrium profiles of normalized magnetic field and resistivity.

where a is the characteristic width of the current sheet. The resistivity of the current sheet is

$$\frac{\eta}{\eta_{ref}} = \cosh^2\left(\frac{y}{a}\right) \tag{62}$$

which satisfies the equilibrium induction equation with no flow. The resistivity has a minimum in the middle of the current sheet ($y = 0$), and the magnetic field vanishes at $y = 0$ and is positive for $y > 0$ and negative for $y < 0$. See Figure 29 for the equilibrium profiles.

3.4.2 Linear Analysis

For the linear analysis, we begin with the incompressible, resistive MHD equations. We assume a variation of the perturbations of the form

$$f = f(y,t)e^{ikx}. \tag{63}$$

The perturbation equations yield a pair of coupled, linear differential equations.[21]

$$\frac{\partial \Psi}{\partial t} = \eta\left(\frac{\partial^2 \Psi}{\partial y^2} - \alpha^2 \Psi\right) - Fw \tag{64}$$

$$\frac{\partial}{\partial t}\left(\frac{\partial^2 w}{\partial y^2} - \alpha^2 w\right) = \alpha^2 Lu^2\left[F\left(\frac{\partial^2 \Psi}{\partial y^2} - \alpha^2 \Psi\right) - \frac{\partial^2 F}{\partial y^2}\Psi\right] \tag{65}$$

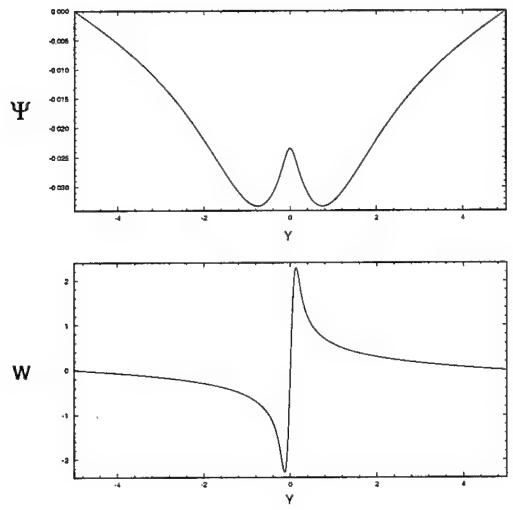


Figure 30: The eigenfunctions for $Lu = 10^3$ and $\alpha = 0.5$.

where Lu is the Lundquist number and

$$\Psi \equiv \frac{B_{y1}}{B_o} \tag{66}$$

$$w \equiv -ik\tau_r v_{x1} \tag{67}$$

$$\alpha = ka \tag{68}$$

$$\tau_r = Lu\tau_A \tag{69}$$

This coupled pair of PDE's are solved numerically using an implicit finite difference formulation. The eigenfunctions for $Lu = 10^3$ and $\alpha = 0.5$ are shown in Figure 30. The growth rates have also been found analytically.[19] For the pure symmetric tearing mode the growth rate is given by

$$\gamma = 0.954(1 - \alpha^2)^{4/5} \left(\frac{Lu}{\alpha}\right)^{2/5}. \tag{70}$$

For values of Lu greater than 500, the numerically calculated and analytical linear growth rates agree to within a few percent.

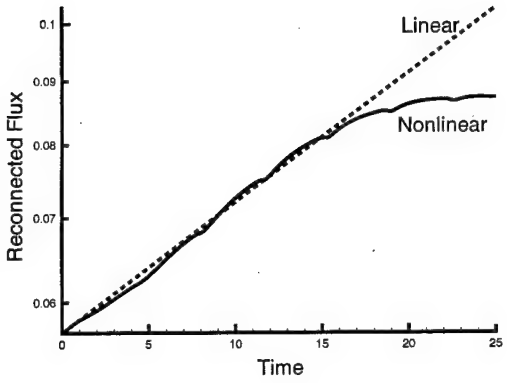


Figure 31: The linear and non-linear evolution of the reconnected flux.

3.4.3 Non-Linear Analysis

A planar current sheet is initialized in the code, and a perturbation of the same form as the linear eigenfunction is superimposed. The growth rate is determined from the amount of reconnected flux at $y = 0$. The evolution of the non-linear perturbation is shown in Figure 31. The result from the linear analysis is also shown. The non-linear growth matches the linear prediction during early development of the instability, but during late time the instability saturates due to non-linear effects. Magnetic flux contours are shown in Figure 32 which show the magnetic island formation of the non-linear instability.

3.5 HIT Injector

One of the first applications for the new code will be to simulate the HIT experiment. The experiment is shown schematically in Figure 33. The geometry is toroidal, but only a single slice in the poloidal plane is pictured. HIT is a low aspect ratio tokamak that uses helicity injection to produce toroidal current. Gas is puffed into the injector and then a series of capacitor banks are discharged across the electrodes to form the plasma and interact with the applied magnetic fields to push the plasma into the confinement region, where the tokamak plasma is formed. The full simulation will require three-dimensional, multiblock capability, which the code does not yet have. However, the injector portion of the experiment can be modeled as a single

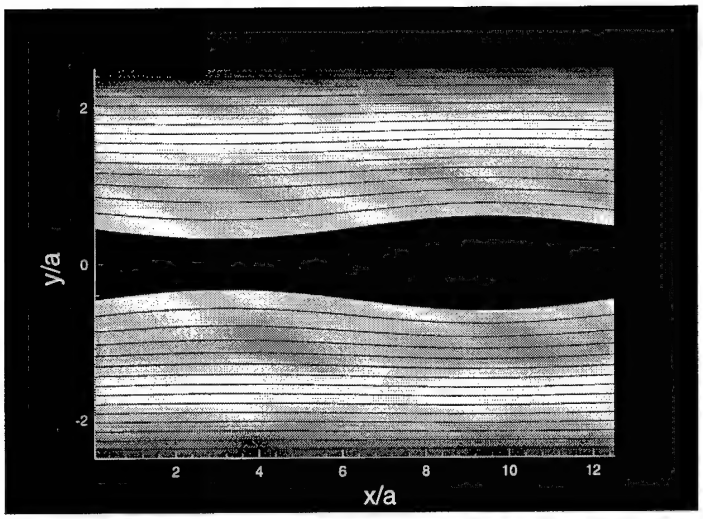


Figure 32: Flux contours of the developed non-linear instability.

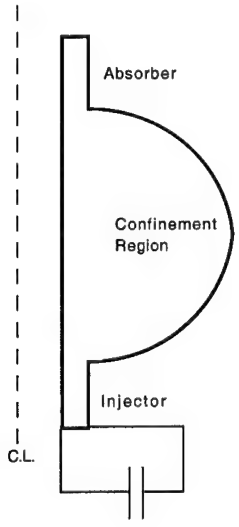


Figure 33: Schematic of the HIT plasma experiment.

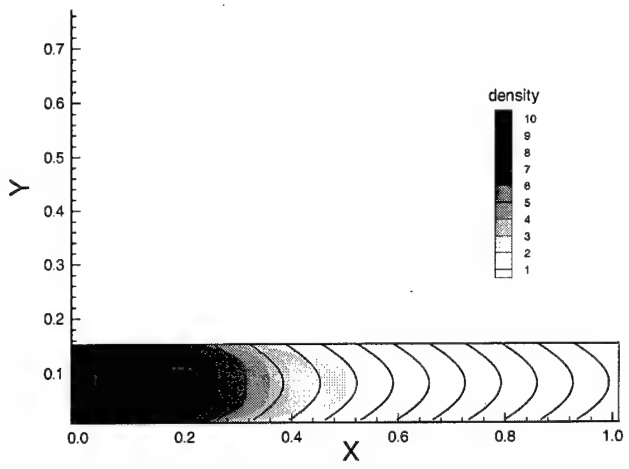


Figure 34: Results of two-dimensional simulation of HIT injector. Plot shows density contours and poloidal magnetic field lines.

block.

For the first simulation we made the further simplification of solving the two-dimensional problem rather than the true cylindrical problem. We chose an initial poloidal bias field that is uniform throughout the injector with $B_y = 0.001T$ and $B_x = 0$, where the x direction is up (toward the confinement region) in Figure 33. The initial toroidal (out of plane) field was zero in this case, and a current of $30kA$ was applied across the electrodes (at the bottom boundary in Figure 33). Plasma was placed at the bottom boundary with a density ten times higher than the initial background density. The Lundquist number was 1000. A grid with 44 cells in the x direction and 12 points in the y direction was used.

The results of the simulation after ten Alfvén times are shown in Figure 34. Contours of density and the in-plane magnetic field lines are plotted. The current at the left boundary ($x = 0$) induces out-of-plane magnetic field that results in a $\mathbf{j} \times \mathbf{B}$ force that brings in plasma from the left and pushes the plasma to the right (in the x direction) towards the confinement region. The contours of density show the higher density plasma being carried into the domain. In addition, the initially straight field lines are stretched as the plasma flows across them. However, the density contours do not overlay

with the field lines as they would in the limit of zero resistivity. This is consistent with the relatively low Lundquist number for this simulation.

3.6 Three-Dimensional Magnetic Relaxation

We have performed a preliminary study of three-dimensional magnetic relaxation in toroidal configurations using the code. These configurations are similar to the previous compact toroid experiment (MARAUDER) at the Phillips Laboratory.[23] The major exception is that the geometry used here is a toroid with a square cross section and smaller aspect ratio. An article describing this application will be published by the Maui High Performance Computing Center (MHPCC) as a HPC Success Story.

The plasma is initialized with uniform density and pressure and a core of purely toroidal field surrounded by purely poloidal field. This corresponds to a current sheet around the toroidal field. The fields strengths are adjusted so the forces balance at the current sheet. In toroidal coordinates, the toroidal field is

$$B_t = B_o \frac{R_o}{R}, \tag{71}$$

and the poloidal field is

$$B_p = B_o \left(\frac{R_o}{R}\right)\left(\frac{r_o}{r}\right), \tag{72}$$

where R is the major radius, r is the minor radius, and B_o is maximum magnetic field value. Because the plasma geometry has a square cross section, the poloidal field beyond $r = a/2$ was set to zero to satisfy the divergence-free condition on the magnetic field.

We know the final plasma configuration based on energy minimization. This configuration is called a Taylor state.[24] Taylor theorized that the magnetic field will rearrange or relax through tearing and reconnection to arrive at the lowest energy state while maintaining the total magnetic helicity. Helicity is the amount of magnetic flux linkage. For this geometry, the Taylor state has an analytical form.

$$B_r = B_o k_z \left[\mathrm{J}_1(k_r r) - \frac{\mathrm{J}_1(\lambda)}{\mathrm{Y}_1(\lambda)}\mathrm{Y}_1(k_r r)\right]\cos(k_z z) \tag{73}$$

$$B_\theta = B_o \sqrt{k_r^2 + k_z^2}\left[\mathrm{J}_1(k_r r) - \frac{\mathrm{J}_1(\lambda)}{\mathrm{Y}_1(\lambda)}\mathrm{Y}_1(k_r r)\right]\sin(k_z z) \tag{74}$$

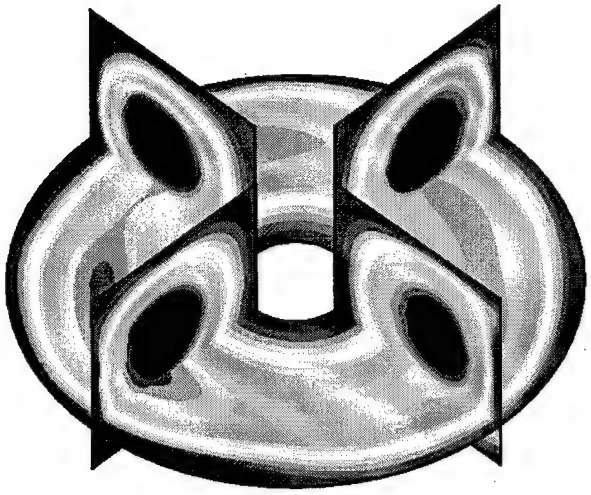


Figure 35: Contours of poloidal flux showing the toroidal mode structure of a relaxing compact toroid with a magnetic Reynolds number of 10^4 after 10 Alfvén transit times.

$$B_z = -B_o k_r \left[\mathrm{J}_0(k_r r) - \frac{\mathrm{J}_1(\lambda)}{\mathrm{Y}_1(\lambda)} \mathrm{Y}_0(k_r r) \right] \sin(k_z z) \tag{75}$$

where $\mathrm{J_m}$ and $\mathrm{Y_m}$ are the ordinary Bessel functions, and k_r, k_z, and λ are chosen to satisfy the boundary conditions.

While Taylor's theory agrees with most experimental data, the mechanism for relaxation is not understood. If the relaxation involves global tearing and reconnection, the process may destroy the plasma. The simulations that we performed are providing great insight into the exact mechanisms and their dependence on the magnetic Reynolds number.

The simulation results for a compact toroid with a magnetic Reynolds number of 10^4 and an aspect ratio of 1.5 are shown in Figure 35 after the plasma has evolved for 10 Alfvén transit times. The formation of a toroidal perturbation with a primary mode number of 3 can be seen in the contours of the poloidal flux. Furthermore, the mode is localized on a magnetic surface ($q = 1$). Interestingly the perturbation saturates and does not destroy the plasma. This mode is a strong candidate for the mechanism responsible for magnetic relaxation.

When the magnetic Reynolds number is decreased, the toroidal mode structure weakens and the simulations are almost axisymmetric. This be-

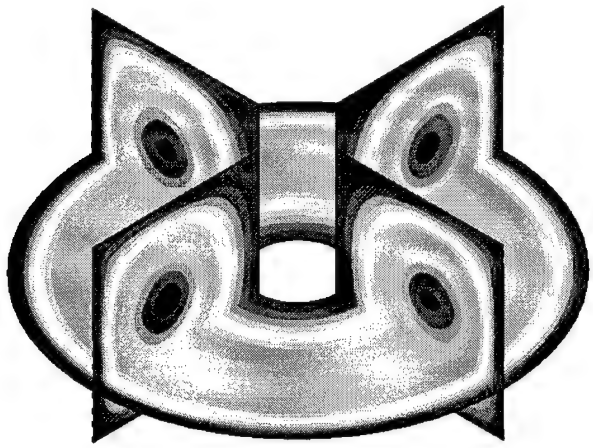


Figure 36: Contours of poloidal flux showing only a weak toroidal mode structure for a relaxing compact toroid with a magnetic Reynolds number of 10^3 after 10 Alfvén transit times.

havior is due to the larger resistive diffusion that occurs at the lower magnetic Reynolds numbers. Instead of magnetic tearing and reconnection, the magnetic fields merely diffuse through each other in a symmetric manner. Results are shown in Figure 36 for a compact toroid with a magnetic Reynolds number of 10^3 and an aspect ratio of 1.5. The results corresponds to the parameters of a cold plasma.

Other simulations were performed at larger aspect ratios. The larger aspect ratios simulations showed a similar mode structure but with a higher toroidal mode number. This phenomena is still under investigation and will be submitted for publication.

3.7 Nonlinear Tilt Instability in the Spheromak

There is a renewed interest in "alternative" plasma confinement concepts. One of these concepts is the spheromak which is a toroidal plasma confinement concept where no materials such vacuum vessels or magnetic field coils link the toroid.[25] The Directed Energy branch of the Air Force Research Laboratory is investigating the use of spheromak plasmas as a target onto which a flux conserving liner will be imploded. This approach may provide a high temperature and high pressure plasma to study magnetized target fusion. Spheromaks were first studied by astrophysicists. They are inter-

esting since they are force-free, simple structures with closed flux surfaces towards which space plasmas tend to evolve.

Plasma stability is a major issue for spheromaks. Stability of spheromaks has been first studied by Rosenbluth and Bussac in a spherical configuration.[26] Using linear theory for a force-free equilibrium ($\mathbf{j} = \lambda \mathbf{B}$, where λ is independent of position), they showed that an oblate spheromak is stable against all internal modes if surrounded by a closed fitting conducting shell. The plasma is unstable if the boundary is prolate.

Finn and Manheimer[27] and Bondeson *et al.*,[28] studied the tilt instability of spheromaks in a cylindrical geometry. The tilt instability is a relaxation to a minimum energy state during which the magnetic axis of the spheromak tilts. In both papers cited above the authors used linear theory and found that for aspect ratios (L/R) less than 1.67 the spheromaks in cylindrical geometry are stable to tilt. However, some experiments showed that oblate spheromaks still tilted.[29]

The goals of our study were to validate the code against theoretical and experimental results obtained for a tilting prolate spheromak and understand why oblate spheromaks tilt.[30] To test our code we tried to match the growth rate obtained with a linear stability code described by Shumlak *et al.*[31] A small perturbation to the spheromak equilibrium should grow initially at the linear growth rate. Eventually nonlinear effects would saturate the growth of the mode.

For our simulation we selected an aspect ratio $L/R = 3$ where linear theory showed that the growth rate is maximum for spheromaks with β between 0 and 6% where $\beta = \frac{p_{max}}{B_{\theta max}/2\mu_0} \times 100\%$. The normalized growth rate obtained was $\gamma \tau_A = 0.20015$.

The nonlinear simulation with our code used a non-simply connected grid made of ten blocks, each with $15 \times 15 \times 20$ cells. (See Figure 37.) This particular type of grid was chosen since it has only four pairs of cells with high aspect ratio. Alternative grid options are a pie slice grid which has high aspect ratio cells all around the circumference and has a singular point at the axis and a distorted square grid which has high aspect ratio cells around the circumference. These alternative grids are shown in Figure 38.

The initial magnetic field is obtained from the force-free equilibrium.

$$
\begin{aligned}
B_r &= -k_z \mathrm{J}_1(k_r r)\cos(k_z z) \\
B_\theta &= \lambda_0 \mathrm{J}_1(k_r r)\sin(k_z z) \\
B_z &= k_r \mathrm{J}_0(k_r r)\sin(k_z z)
\end{aligned}
$$

where $k_r R = 3.832$, $k_z L = \pi$ and $\lambda_0 = \sqrt{k_r^2 + k_z^2}$. Uniform density and

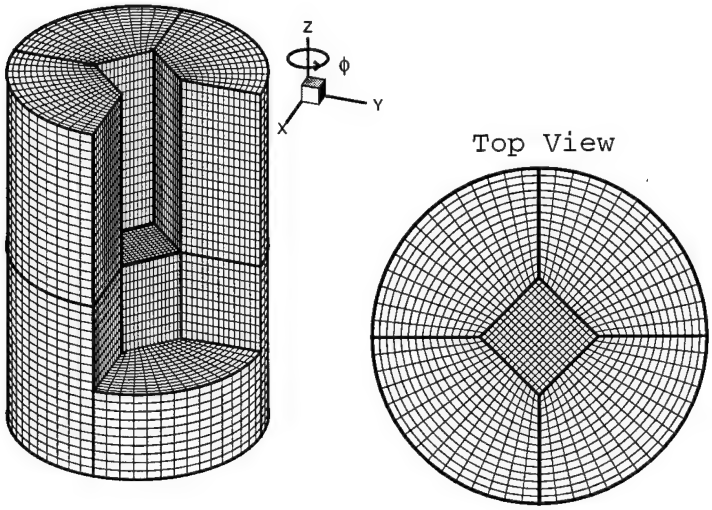


Figure 37: The ten block grid used for the spheromak simulation. Some of the blocks has been removed for illustration.

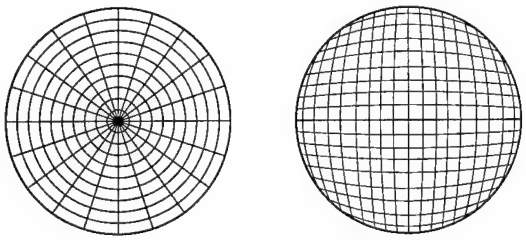


Figure 38: Two alternate grids that can be used for simulations of cylindrical configurations. **(a)** The pie slice grid has large aspect ratio cells along circumference and a singularity at the axis. **(b)** The distorted square grid has high aspect ratio cells along the circumference.

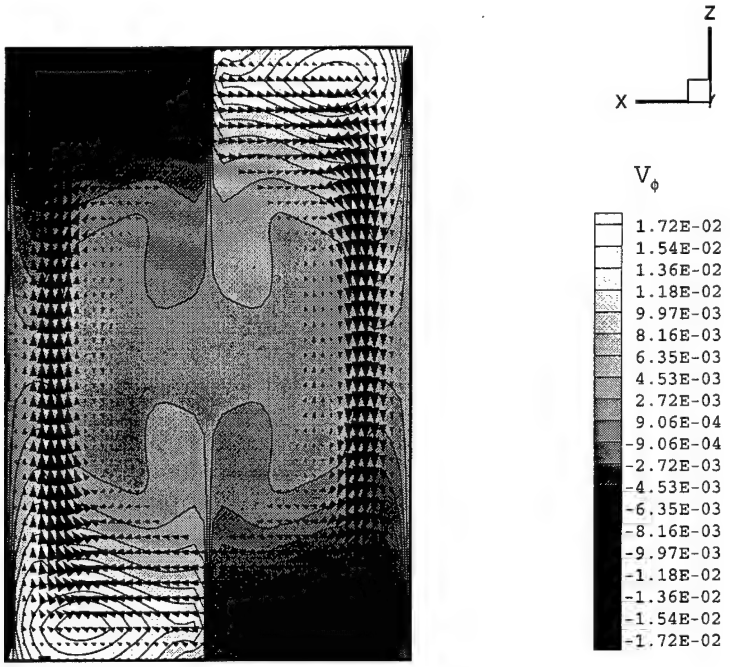


Figure 39: Initial velocity field. Contours represent the magnitude of the toroidal velocity component and the vectors represent the poloidal velocity. Velocity field is normalized with respect to the Alfvén speed.

pressure profiles are initialized to give $\beta = 8\%$. The initial perturbation is the velocity field obtained from the linear code and interpolated to the three dimensional grid. Figure 39 shows the initial velocity field.

Our first goal was to match the growth rate obtained from the linear code at early time. Since our code is nonlinear, the size of the initial perturbation is critical. We found that if the initial perturbation is too large ($v_{max}^{t=0} > 12\% v_A$) then the spheromak structure is destroyed. If the initial perturbation is too small the algorithm produces flows which are larger than the initial perturbation. These spurious flows were the result of the generalized coordinate formulation which are being corrected by using the unaligned finite volume formulation described in Section 2.2.8.

An initial velocity perturbation causes the spheromak to tilt as shown in Figure 40. When the tilt instability saturates, the plasma axis is not

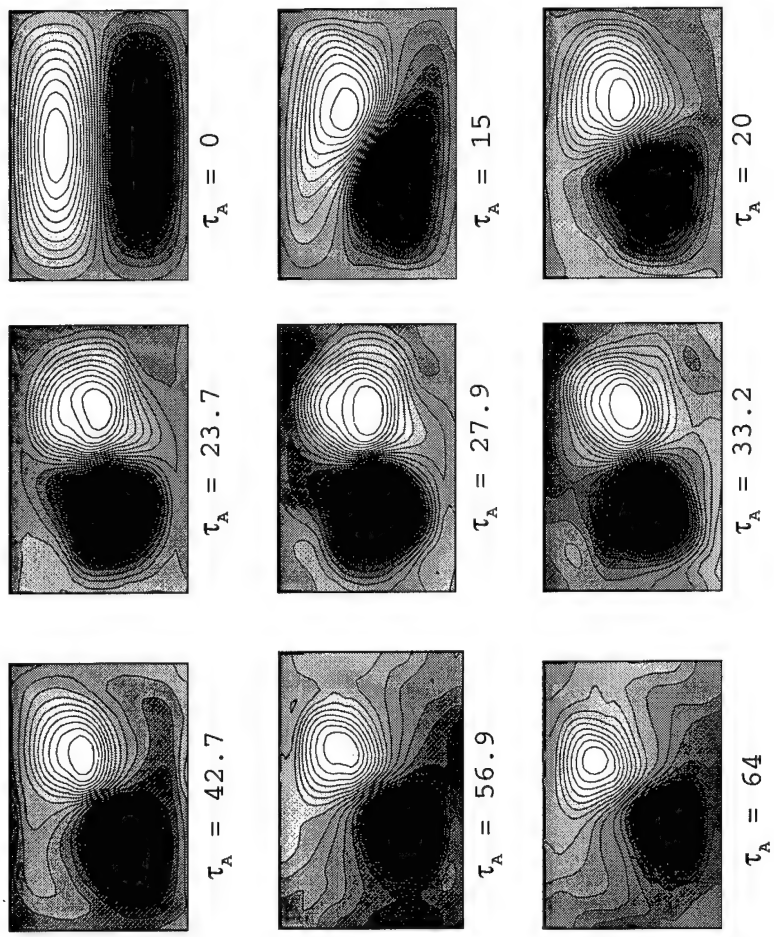


Figure 40: Contours of toroidal magnetic field B_θ through a cross-section of the spheromak at $\theta = 90^o$. White contours represents positive values and black negative values.

perpendicular to its original orientation. The final state is oriented so that the plasma has expanded into the corners of the flux conserver which further minimizes the magnetic energy. Figure 41 shows the growth of the kinetic energy with time. The linear growth rate has also been plotted for comparison. The agreement in these improved results indicates the success of the improved algorithm and of the unaligned finite volumes implementation.

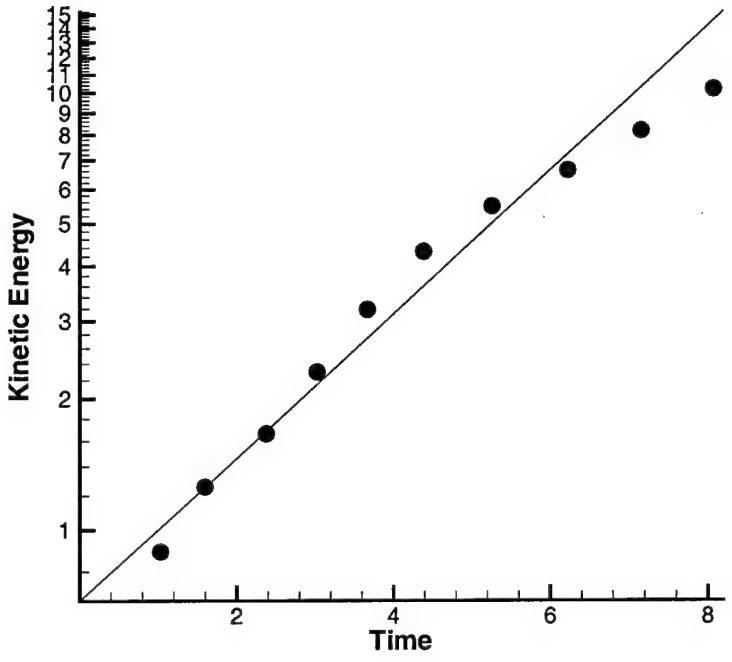


Figure 41: Evolution of the kinetic energy of the spheromak as a function of normalized time. The solid line is the linear growth rate.

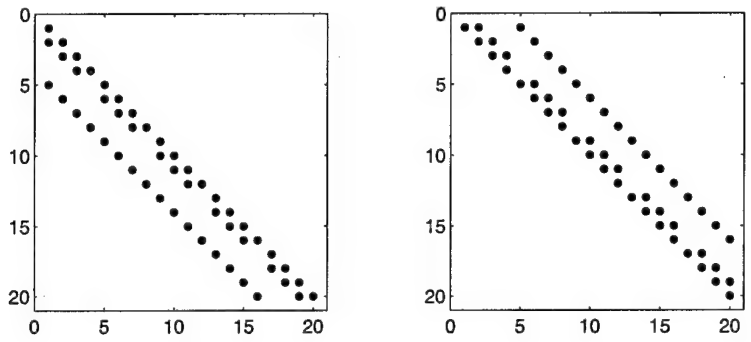


Figure 42: The 20×20 lower (a) and upper (b) tridiagonal block matrices for the LU-SGS algorithm with a grid of 4×5 cells.

4 Parallel Computer Implementation

We have investigated strategies for implementing the algorithm on parallel architectures. The first of the following sections describes the simplest approach, which parallelizes the LU-SGS algorithm in a point-wise manner. This proved to be too fine-grained to be efficient, so we have opted for a domain decomposition approach which is described in the second section. The third section describes the implementation of this method on the MHD solver.

4.1 Fine-Grain Parallelization

The LU-SGS algorithm involves a double sweep of the computational domain. The forward (predictor) sweep solves a lower tridiagonal block matrix for the entire computational domain. The backward (corrector) sweep solves an upper tridiagonal block matrix. Figure 42 shows the form of the lower and upper block diagonal matrices for the case of a 4×5 grid. Because of the lower-upper form of the matrices, the solutions at grid cells along a line of constant $i + j$ are independent.

The simplest parallel implementation is to decompose the domain into its component cells, distribute the grid cells over the processors of the parallel computer, and treat each cell as residing on a different processor. This approach exploits the independence of the solutions of the cells on lines of constant $i + j$. Communication between the cells provides the necessary synchronization. For these tests, we used the Parallel Virtual Machine

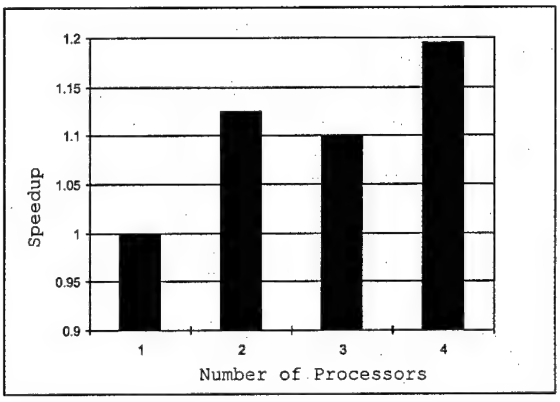


Figure 43: The parallel speedup for a problem with constant size grid using a fine-grain parallelization approach.

(PVM) communication library which was developed at Oak Ridge National Laboratory.[32] PVM allowed us to connect a network of four DEC Alpha workstation and use them as our parallel computer.

To determine the parallel effectiveness, we measured the speedup obtained when a problem grid of constant size was evenly distributed onto an increasing number of processors. Speedup is defined as the time required to find the solution with n processors divided by the time with one processor. For perfectly parallel implementations, the speedup would be equal to the number of processors. Any communication time and processor synchronization decreases the speedup.

We used a 4×4 grid and varied the number of processors from one to four. While this was a small size problem, it was sufficient to test the parallel implementation. The speedup results are shown in Figure 43. Some speedup can be seen; however, the amount is unsatisfactory.

The low efficiencies indicate that the simplest approach for parallel implementation of the LU-SGS algorithm is inadequate. The results are not surprising since the grain of parallelization in this approach is too fine and requires excessive communication. The number of grid cells in practical applications will be much greater than the number of processors. This suggests dividing the domain into a number of large blocks, so that the grid cells within a block are located on the same processor (and memory) and do not need to communicate through message passing.

4.2 Coarse-Grain Parallelization

In this section, we describe the coarse-grain parallelization of the MHD solver and the performance of this approach applied to a real problem.

The algorithm was parallelized using the domain decomposition technique (DDT). This technique is based on the simple idea of "divide and conquer" The integral form of a general conservation law is

$$\frac{\partial}{\partial t}\int_{\Omega} dV\ Q + \oint_{\Sigma} dS \cdot F(Q) = \int_{\Omega} dV\ S(Q), \tag{76}$$

where Ω is the domain and Σ is the boundary of Ω. Q is the vector of conserved variables, $F(Q)$ is the flux of the conserved variables, and $S(Q)$ is the vector of source terms. By splitting the domain Ω into p subdomains such that

$$\Omega = \bigcup_{i=1}^{p} \Omega_i, \tag{77}$$

one can replace eqn(76) with a set of p conservation equations applied on the subdomains Ω_i.

$$\frac{\partial}{\partial t}\int_{\Omega_i} dV\ Q + \oint_{\Sigma_i} dS \cdot F(Q) = \int_{\Omega_i} dV\ S(Q), \quad i = 1, 2, ..., p \tag{78}$$

Each of these discretized equations is solved by a single processor. Each processor uses the boundary values copied from neighboring subdomains.

4.2.1 Domain Decomposition

To abstract the computer architecture, we assume that a set of p processors can be assigned to run the code and that these processors implement a message passing system. For simplicity the original domain is assumed to be a square of size $n \times n$.

The 2-D version of the algorithm was parallelized. There are two techniques available for the decomposition of 2-D domains, the strip decomposition and the patch decomposition which are shown in Figure 44.

Strip decomposition is implemented by dividing the original domain in subdomains of $n \times \frac{n}{p}$, and it might be thought of as a 1-D decomposition. With strip decomposition each subdomain needs to exchange data with two neighbors except the subdomains at the boundaries of the original domain which communicate with only one neighbor.

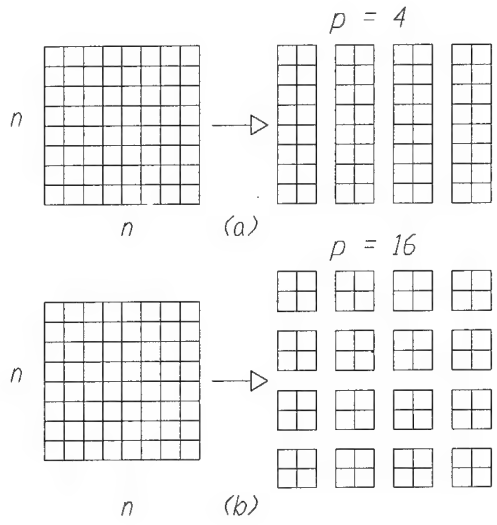


Figure 44: (a) Strip decomposition and (b) patch decomposition of a 2-D domain.

The communication time for an interior subdomain was defined by Zhu [33] as

$$T_{D_{21}} = 2(\sigma + 8\beta n) \tag{79}$$

where σ is the communication start-up time, β is the time required to send one byte of data, and the 8 means that the data are represented as double precision variables (their size is eight bytes).

Patch decomposition is implemented by dividing the original domain in $\frac{n}{\sqrt{p}} \times \frac{n}{\sqrt{p}}$. With this method each processor has to communicate with four neighbors unless it is situated on the boundaries of the original domain and it has two or three neighbors. For simplicity it is assumed that p is an even square number and n is evenly divisible by $\sqrt{p}$. The communication time for an internal subdomain is

$$T_{D_{22}} = 4(\sigma + 8\beta\frac{n}{\sqrt{p}}). \tag{80}$$

For a fixed grid size, $T_{D_{22}}$ decreases with the number of processors since in eqn (80) the number of processors appear at the numerator. In contrast, $T_{D_{21}}$ stays constant with the number of processors.

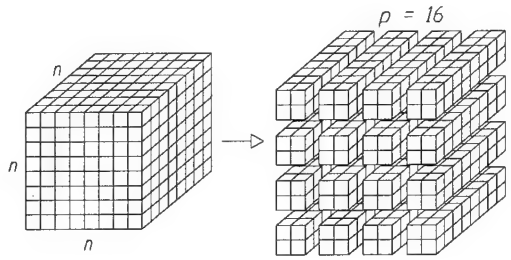


Figure 45: Column decomposition of a 3-D domain is an immediate extension of the patch decomposition of 2-D domains.

This made the patch decomposition an obvious choice for our implementation. The technique will also provide a straightforward extension to the column decomposition of 3-D domains (see Figure 45).

4.2.2 Implementation of the Patch Decomposition

The programming model used for the implementation was single program multiple data (SPMD). Each processor runs the same code on the data corresponding to its subdomain. One processor has to perform the domain decomposition and send the data to the other processors. This processor was designated as the *main task*.

Assuming that there are p processors available for running the code they can be arranged in a processor grid of $p_r \times p_c = p$ where p_r is the number of rows and p_c is the number of columns. The size of the original computational domain is $m \times n$. It is possible to have subdomains of equal sizes only if m and n are evenly divisible with p_c and p_r respectively. The domain decomposition was implemented such that some processors receive an extra row or extra column if m and n are not divisible by p_c and p_r.

Physical coupling of the subdomains is accomplished by the exchange of internal boundaries. A processor sends the data from the cells next to its boundaries to the neighboring processors if they exist. The receiving processor assigns the received data to the cells of its respective boundaries. If a processor does not have a neighbor in a certain direction the boundary conditions are applied to that boundary. Since the algorithm uses a five-point stencil only one row/column needs to be exchanged.

4.2.3 Message Passing

One of the goals of the project is to develop a portable code. A first step in assuring the portability was to use a message passing system commonly available on parallel supercomputers and on workstation clusters. This system is the Message Passing Interface (MPI)[13], which was adopted as a standard in May 1994 by industry and academia. Hardware and software vendors' implementation of MPI provides parallel program developers with a consistent set of subroutines callable from FORTRAN77 and C. In our code we made use of the basic point-to-point communications subroutines and global communications subroutines. The point-to-point communication subroutines were used for the domain decomposition and boundary exchange while the global communication subroutines were used for convergence checking. All message passing systems (PVM, MPL) support point-to-point and global communications subroutines so that by using only the basic set we provided for a facile portability to systems not supporting MPI.

4.2.4 Load Balancing

The load balancing for this code is performed by distributing an approximately equal number of cells to each processor. This is accomplished by the *main task* during the domain decomposition phase. Since the number of floating point operations performed by each processor is the same, a static domain decomposition is sufficient to ensure that the processors have an equal share of the computing load. If the code takes were to allow for time-dependent ionization or other localized phenomena which require additional operations in a limited region of the computational domain, then a dynamic load balancing procedure may be necessary. A simple algorithm for dynamic load balancing is the masked multiblock described by Borrelli *et al.*[34] We will implement the masking algorithm in future versions of the code if it becomes necessary.

4.2.5 Results

In order to measure the performance of the code we applied the parallel version to the plasma gun problem described in Section 3.3. The parallel version was checked against the sequential version, and both produced identical results.

There are two criteria generally used for the performance analysis of parallel codes: (1) the speedup $S_p = T_{seq}/T_p$ and (2) the efficiency $E_p = S_p/p$, where T_{seq} is the time needed for the best sequential algorithm to

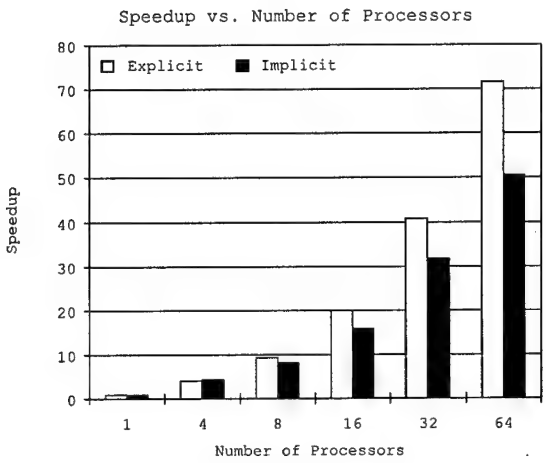


Figure 46: Fixed grid (400×80) speedup results. Note the superlinear speedup of the explicit mode.

complete the task and T_p is the time needed for the parallel algorithm run on a number of p processors to complete the same task. Note that the definition of speedup used here is more rigorous and meaningful than the one commonly used since it is based on the sequential version and not the parallel version on one processor.

We ran the parallel code on the IBM SP2 with a fixed grid of size 400×80 on a processor pool of varying size: 4, 8, 16, 32 and 64 processors. The speedup for the explicit and implicit modes is shown in Figure 46. As expected the speedup increases with the number of processors assigned to run the code. For the explicit mode the speedup is superlinear, which seems to contradict Amdahl's law

$$S_p = \frac{T_{seq}}{T_{communication} + \frac{\sum_{i=1}^{p} T_{computation,i}}{p}}. \tag{81}$$

Assuming that no time is used for communication and that the sum of the computation time for all processors is equal to the sequential computation time, the maximum speedup is linear (for p processors the speedup is p). However, Amdahl's law does not take in consideration the architecture of the system used, in particular the cache effects. On the IBM RS/6000 machines, which constitute the nodes of the SP2, the data is passed from the main memory to the CPU through a data cache. A data cache miss involves a delay of eight CPU cycles while the data in the cache can be accessed

in one cycle[35]. Noting that an add and multiply operation (a FLOP) takes one CPU cycle the conclusion is that a data cache miss decreases the performance significantly. By increasing the number of processors in the pool and keeping the overall problem size constant, we reduced the amount of data assigned to a processor. Its data cache could hold more data thus reducing the number of cache misses and improving the performance, which explains the superlinear speedup. The same behavior was observed by Michl *et al.*, on a cluster of IBM RS/6000/500 workstations.[36]

The speedup for the explicit mode is higher than that for the implicit mode because the implicit mode is the more computationally intensive and is, therefore, less sensitive to cache misses. One has to be careful when comparing the results presented in Figure 46 since the number of iterations until convergence is reached for the implicit mode depends on the number of processors used.

The trend of the speedup shows an increasing slope for both explicit and implicit modes which indicates that the code is far from communication saturation. Saturation occurs when the time spent on communications becomes comparable with the computation time. If the number of processors is increased and the size of the subdomains becomes smaller, each processor will have fewer computations to perform, but the total time spent in exchanging the data on the boundaries will increase. The total time spent for boundary exchange can be found using the formula for the communication time for an internal subdomain [eqn(80)] and multiplying it with the number of processors in a pool p,

$$T_{bdry\ exch} = pT_{D_{22}} = 4(\sigma p + 8\beta n\sqrt{p}). \tag{82}$$

The total time spent on boundary exchanges varies proportionally with p.

For the processor pools with a non-square number of processors we have run the code on grids organized as $p_r \times p_c$ and the transpose $p_c \times p_r$, so that the number of row cells versus column cells changed. The results showed that a decomposition whose subdomains have longer rows performs better than one with longer columns. This is consistent with the data cache misses that were observed previously. An improvement of 20–30% in the measured speedup was obtained by modifying the domain partitions. It should be noted that this result is particular to IBM architecture, and the dependency of the obtained speedup on domain decomposition will vary on other architectures. The speedup results shown in Figure 46 for 8 and 32 processors have been averaged.

In order to eliminate the cache effects from the performance analysis we ran the code on grids that scaled with the number of processors. The size of

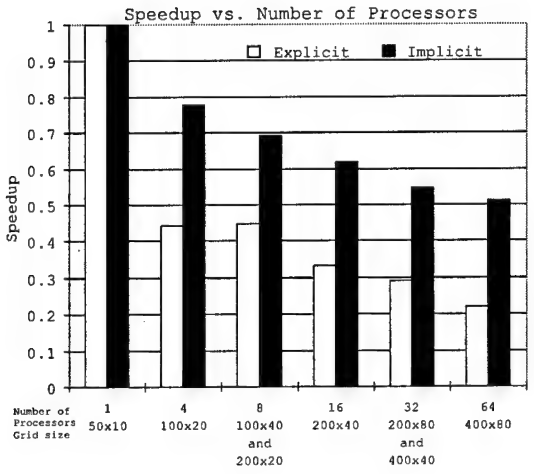


Figure 47: Scaled grid (50×10 per processor) speedup results.

the grid on each processor remained constant. As the number of processors was increased, the grid increased proportionally. The speedup results are presented in Figure 47. Again note that the speedup is measured relative to the sequential version of the code and not the parallel version run on a single processor.

The speedup for a perfectly parallel code for the scaled grid is unity for any number of processors. Our results show a speedup that is less than unity and it decreases with the number of processors. This is an expected result since the total communication time increases with the number of processors. Since the slope of the speedup is gradual and it appears to flatten, we conclude that the parallel code performs satisfactorily on scaled grids.

4.3 Parallel Performance on Windows NT

As described in the previous section we have ported our code to several parallel architecture platforms. The ports have been simple since we have committed to using standard Fortran90 language and MPI for our message passing.[13] Our code now also runs on clusters of Windows NT personal computers. The parallel speed up in performance has been quite impressive even for the Windows NT machines. Table 1 shows the processor utilization for a "scaled problem" parallel benchmark which was run on the IBM SP2

No. of Processors	Parallel Speedup	
	IBM SP2	Windows NT
1	1.00	1.00
4	0.78	0.89
8	0.69	
16	0.62	
32	0.55	
64	0.52	

Table 1: Parallel speedup for a "scaled problem" benchmark on the IBM SP2 and a quad-processor Windows NT workstation.

at the Maui High Performance Computing Center (MHPCC) and a quad-processor Intel workstation running Windows NT. The ratio of problem size to number of processors is constant for a scaled problem.

Moving to the Windows NT platform also allowed us to write a graphical user interface (GUI) for the code. See Figure 48. The GUI allows the user to visualize the multiblock grid layout and prompts the user for initial and boundary conditions. The interface provides visual and logical confirmation that the problem is setup correctly before the user commits to a compute intensive simulation.

5 Professional Interactions

5.1 Project Personnel

The personnel who have been directly involved in this project are listed below.

Name	Position
Uri Shumlak	Research Assistant Professor
D. Scott Eberhardt	Associate Professor
Thomas R. Jarboe	Professor
Byoungsoo Kim	Research Associate
Julian Becerra-Sagredo	Graduate Student
Ogden S. Jones	Graduate Student
R. Scott Raber	Graduate Student
David Taflin	Graduate Student
Bogdan Udrea	Graduate Student
Ward Vuillemot	Graduate Student

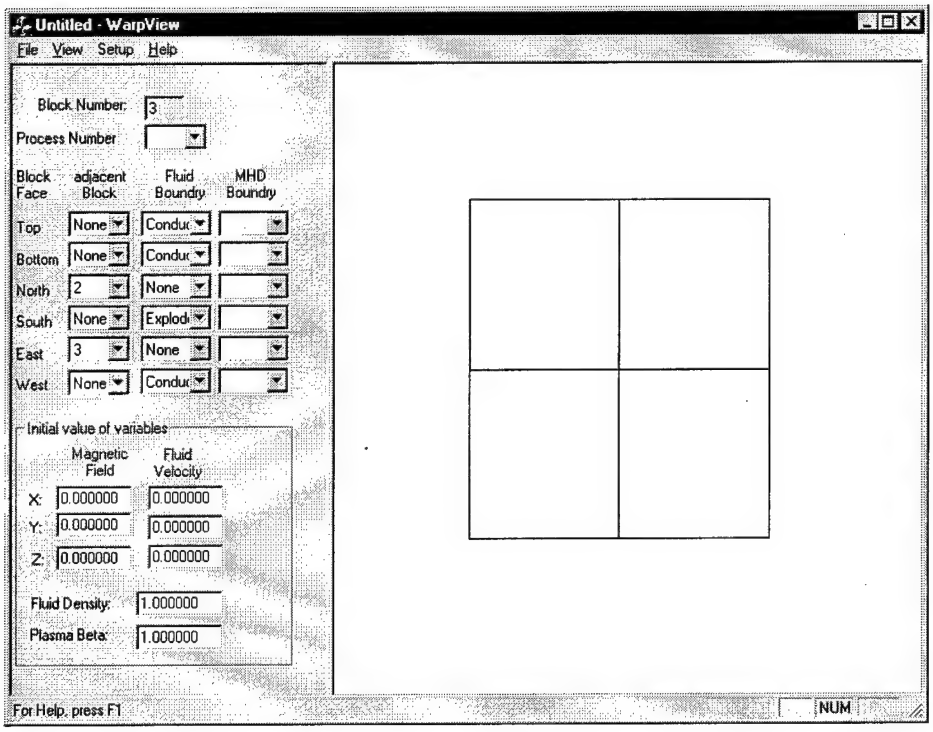


Figure 48: Screen shot of the GUI which controls our code on Windows NT platforms.

5.2 Collaborations

5.2.1 Air Force Research Laboratory

Dr. Robert Peterkin, Jr. of the Electromagnetic Sources Division of the Air Force Research Laboratory at Kirtland AFB on three-dimensional multigrid algorithms for MACH3, development of a parallel PIC (particle in cell) code for microwave simulations, and stabilization of the Rayleigh-Taylor instability in solid liner implosions by introducing a sheared axial flow. Knowledge we developed in the area of relaxation schemes was implemented into the ICEPIC code to make a 3-D Poisson solver. The solver was needed to determine electric field concentration on a high power microwave source. The collaboration occured in person during March and April. Several phone and email discussions took place throughout the year.

5.2.2 Sandia National Laboratories

Dr. Norm Roderick of the Pulsed Power Sciences Center at Sandia National Laboratories on the uses of sheared axial flows to stabilize z-pinch implosions. This was an ongoing collaboration that resulted in the publication listed in the previous section.

5.2.3 University of Washington

Prof. Scott Eberhardt of the Aeronautics and Astronautics Department and Prof. Randy LeVeque of the Applied Math Department on approximate Riemann solvers and their applications to multidimensional problems. We have regular discussions on a weekly basis.

5.3 Publications

A journal article describing our algorithm has been published in the *Journal of Computational Physics.* The title is "An Implicit Scheme for Nonideal Magnetohydrodynamics" by O. S. Jones, U. Shumlak, and D. S. Eberhardt.[14] The citation is Journal of Computational Physics **130,** 231 (1997). Another journal article which describes the use of sheared flows to stabilize the Rayleigh-Taylor instability has been published in *Physics of Plasmas.* This is work that was performed with collaboration at the Air Force Research Laboratory. The title is "Mitigation of the Rayleigh-Taylor Instability by Sheared Axial Flows" by U. Shumlak and N. F. Roderick.[37] The citation is Physics of Plasmas **5,** 2384 (1998).

We have two papers that have been published in conference proceedings. "Physics of the Hall Thruster," U. Shumlak, T. R. Jarboe, and R. A. Sprenger, AIAA 97-3048 (1997), at the 1997 Joint Propulsion Conference in Seattle, Washington. "Non-linear Study of the Spheromak Tilt Instability," B. Udrea and U. Shumlak, AIAA-98-0995 (1998), at the Aerospace Science Meeting in Reno, Nevada.

5.4 Presentations

A paper was presented at the Thirty-Sixth AIAA Aerospace Science Meeting, Reno, Nevada, January 1998. The title was 'Non-linear Study of the Spheromak Tilt Instability," B. Udrea and U. Shumlak. Two papers were presented at the Twenty-Fifth Annual IEEE International Conference on Plasma Sciences, Raleigh, North Carolina, June 1998. The titles were "A Finite Volume Implementation of an Approximate Riemann Solver for MHD," by U. Shumlak and B. Udrea; "Numerical Study of a Magnetic Cumulative Generator," by B. Udrea and U. Shumlak.

Several papers were presented at the Fortieth Annual American Physical Society Meeting of the Division of Plasma Physics, New Orleans, Louisiana, November 1998. The titles were "An Approximate Riemann Solver for MHD Computations on Parallel Architectures," by U. Shumlak, D.S. Eberhardt, B. Udrea, R.S. Raber, and J. Becerra-Sagredo; "Computational MHD on an Intel Based Windows NT Platform," by R.S. Raber, U. Shumlak, and B. Udrea; and "Nonlinear Study of Spheromak Tilt Instability," by B. Udrea and U. Shumlak.

6 Conclusions

The successful development of the three-dimensional advanced implicit algorithm, the implementation of the algorithm on arbitrarily connected multiblock grids, and the practical applications indicate that this project is reaching its objectives. The research from this project has been published in refereed journals and presented at international conferences. Valuable collaborations have been formed with the Air Force Research Laboratory, Sandia National Laboratory, and other universities.

The continuing development of this project will include investigating more powerful implicit matrix inversion methods, adding multi-species (including neutrals), and applying the code to plasma experiments to calibrate the code and gain physical insight into devices that are important to the Air Force.

References

[1] G. H. McCall, J. A. Corder and the USAF Scientific Advisory Board, *New World Vistas, Air and Space Power for the 21st Century,* United States Air Force Document, Summary Volume (December 1995).

[2] U. Shumlak, T. W. Hussey, and R. E. Peterkin, Jr., IEEE Trans. Plasma Sci. **23**(1), 83 (1995).

[3] F. M. Lehr, A. Alaniz, J. D. Beason, L. C. Carswell, J. H. Degnan, J. F. Crawford, S. E. Englert, T. J. Englert, J. M. Gahl, J. H. Holmes, T. W. Hussey, G. F. Kiuttu, B. W. Mullins, R. E. Peterkin, Jr., N. F. Roderick, P. J. Turchi, and J. D. Graham, J. Appl. Phys. **75**, 3769 (1994).

[4] B. A. Nelson, T. R. Jarboe, D. J. Orvis, L. McCullough, J. Xie, C. Zhang, and L. Zhou, Phys. Rev. Lett. **72**, 3666 (1994).

[5] H. P. Furth, J. Kileen, and M. N. Rosenbluth, Phys. Fluids **6**, 479 (1963).

[6] H. Ok and D. S. Eberhardt, "Solution of Unsteady Incompressible Navier-Stokes Equations Using an LU Decomposition Scheme," AIAA-91-1611 (1991).

[7] S. Yoon and A. Jameson, AIAA J. **26**, 1025 (1988).

[8] P. L. Roe, J. Comp. Phys. **43**, 357 (1981).

[9] R. J. LeVeque, *Numerical Methods for Conservation Laws,* Birkhauser Verlag, Boston (1992).

[10] M. Brio and C. C. Wu, J. Comp. Phys. **75**, 400 (1988).

[11] A. L. Zachery and P. Colella, J. Comp. Phys. **99**, 341 (1992).

[12] K. G. Powell, B. van Leer, and P. L. Roe, *Private Communication,* 1994.

[13] Message Passing Interface Forum *MPI: A Message Passing Interface Standard,* May 5, 1994.

[14] O. S. Jones, U. Shumlak, and D. S. Eberhardt, J. of Comp. Physics **130,** 231 (1997).

[15] B. Einfeldt, C. D. Munz, P. L. Roe, B. Sjögreen J. of Comp. Physics **92,** 273 (1991).

[16] N. Aslan and T. Kammash, J. Comput. Phys. **133,** 43 (1997).

[17] D. Harned and Z. Mikic, J. Comput. Phys. **83**, 1 (1989).

[18] P. C. Sleziona, M. Auweter-Kurtz, H. O. Schrade, Int. J. Num. Meth. Eng. **34,** 759 (1992) .

[19] H. P. Furth, J. Killeen, and M. N. Rosenbluth, Phys. Fluids **6**(4), 459 (1963).

[20] H. P. Furth, Phys. Fluids **28**(6), 1595 (1985).

[21] J. Killeen and A. I. Shestkov, Phys. Fluids **21**(10), 1746 (1978).

[22] D. Schnack and J. Killeen, J. of Comp. Physics **35**, 110 (1980).

[23] J. H. Degnan, R. E. Peterkin, Jr., G. P. Baca, J. D. Beason, D. E. Bell, M. E. Dearborn, D. Dietz, M. R. Douglas, S. E. Englert, T. J. Englert, K. E. Hackett, J. H. Holmes, T. W. Hussey, G. F. Kiuttu, F. M. Lehr, G. J. Marklin, B. W. Mullins, D. W. Price, N. F. Roderick, E. L. Ruden, C. R. Sovinec, P. J. Turchi, G. Bird, S. K. Coffey, S. W. Seiler, Y. G. Chen, D. Gale, J. D. Graham, M. Scott, and W. Sommars, Phys. Fluid B **5**, 2938 (1993).

[24] J. B. Taylor, Phys. Rev. Lett. **33,** 1139 (1974).

[25] T. R. Jarboe, Plasma Phys. Controlled Fusion **36,** 945 (1994).

[26] M. N. Rosenbluth and M. N. Bussac, Nucl. Fusion **19,** 489 (1979).

[27] J. M. Finn and W. M. Manheimer, Phys. Fluids **24**(7), 1336 (1981).

[28] A. Bondeson, G. Marklin, Z. G. An, H. H. Chen, Y. C. Lee, and C. S. Liu, Phys. Fluids **24**(9), 1682 (1981).

[29] C. W. Barnes, I. Henins, H. W. Hoida, T. R. Jarboe, G. Marklin, B. L. Wright, and G. A. Wurden, *Conference Record of the 1984 IEEE International Conference on Plasma Science* St. Louis, MO, pp. 50-1.

[30] T. R. Jarboe, I. Hennis, H. W. Hoida, R. K. Linford, J. Marshall, D. A. Platts, and A. R. Sherwood, Phys. Rev. Lett. **45**(15), 1264 (1980).

[31] U. Shumlak, T. K. Fowler, and E. C. Morse, Phys. Plasmas **1**(3), 643 (1994).

[32] Al Geist, A. Beguelin, J. Dongara, W. Jiang, R. Manchek, V. Sunderam, *PVM 3 User's Guide and Reference Manual,* 1994.

[33] J. Zhu, *On the Implementation Issues of Domain Decomposition Algorithms for Parallel Computers* Parallel CFD Conference, 1992.

[34] S. Borrelli, A. Matrone, P. Schiano, *A Multiblock Hypersonic Flow Solver For Massively Parallel Computers,* Parallel CFD Conference, 1992.

[35] IBM Corp., *Optimization/Tuning Guide for XL FORTRAN and XLC.*

[36] T. Michl, S. Wagner, M. Lenke, A. Bode, *Dataparallel Implicit Navier-Stokes Solver on Different Multiprocessors* Parallel CFD Conference 1993.

[37] U. Shumlak and N. F. Roderick, Phys. Plasmas **5,** 2384 (1998).